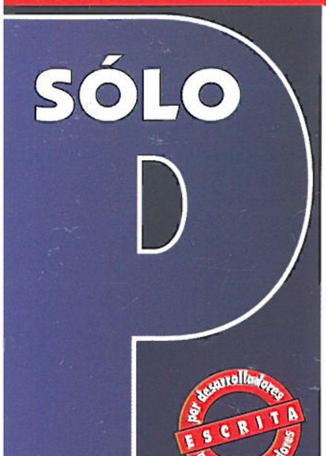
LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO



# PROGRAMADORES

Revista especializada para usuarios de PC

ESCRITA por desarrolladores para desarrolladores

AÑO 3. Nº 26
1250 PTAS.

ARGENTINA 9'50 $
CHILE 3000 $
PORTUGAL 1500$

## LINUX: EL GESTOR DE VENTANAS X-WINDOWS

## WEB: LENGUAJE HTML

## VISUAL BASIC: VARIABLES, CONSTANTES Y MATRICES

CD-ROM DE REGALO:
LIBRERÍAS DE PROGRAMACIÓN PARA WINDOWS 95

### Y además:

- Los servicios DOS
- El protocolo IP
- Cómo programar una Demo

### CURSOS PRÁCTICOS DE:

- Grandes Sistemas
- Visual C++
- Modos X de la VGA
- Redes Locales

# EUSKAL PARTY 96

## EL FESTIVAL DE LA DEMOSCENE

8 413042 303299 00026

TOWER COMMUNICATIONS, S.R.L

IBM D_Mail

Network Management

La tarjeta de respuesta ya ha sido utilizada.

Unix

Parallel Systems

Cliente/Servidor

Internet

Bases de datos relacionales

N
W
E
S

NO se preocupe. Envíenos por correo electrónico su nombre, dirección y función a:
SDOEMEA@UK.IBM.COM
o llámenos al 900 100 400

Objetos

# *Cómo lograr y mantener una posición líder en el vertiginoso negocio del desarrollo de software.*

Para desarrollar programas se necesitan buenas ideas así como los conocimientos y las herramientas necesarios para tranformarlas en soluciones excepcionales. La clave para llegar y mantenerse en una posición destacada es el acceso fácil a la información necesaria. Por eso IBM ha creado D_Mail: un nuevo servicio de información para empresas y profesionales desarrolladores de software.

D_Mail le facilita la más reciente información sobre el impacto potencial de las últimas innovaciones, análisis detallados sobre tecnologías clave, el panorama de oportunidades profesionales para el mundo de las soluciones de software y mucho más. ¡Sin coste para usted!

Sólo tiene que devolvernos cumplimentada la tarjeta adjunta, y nosotros le ayudaremos a trazar el rumbo correcto hacia el futuro.




Soluciones para nuestro pequeño mundo




OCTUBRE 1996. Número 26
**SÓLO PROGRAMADORES es una publicación de Tower Communications**

**Director General**
Antonio M. Ferrer Abelló
**Director Financiero**
Francisco García Díaz de Liaño
**Director de Producción**
Carlos Peropadre
**Directora Comercial**
Carmina Ferrer
**Director de Distribución**
Enrique Cabezas García

**Editor**
Antonio M. Ferrer Abelló
**Coordinador Técnico**
Eduardo Toribio Pazos
**Edición**
Miguel Cabezuelo
**Colaboradores**
Fernando de la Villa, Fernando J. Echevarrieta, Pedro Antón. Juan M. Martín, Luis Martín, José M. Peco, José C. Remiro, Agustín Guillén, Daniel Navarro, Jorge del Río, Enrique De Alarcón, David Aparicio, M. Jesús Recio, Santiago Romero, Miguel Cubas, Vicente Cubas, Carlos Pérez

**Jefe de Diseño**
Fernando García Santamaría
**Maquetación**
Clara Francés
**Tratamiento de Imagen**
María Arce Giménez

**Suscripciones**
Isabel Bojo
**Publicidad en Madrid**
Erika de la Riva
**Redacción, Publicidad y Administración**
C/ Aragoneses, 7
28100 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86
**Publicidad en Barcelona**
Pepín Gallardo
(93) 213 42 29
**Filmación**
Filma Dos S.L.
**Impresión**
G. Reunidas
**Distribución**
SGEL



**Depósito legal:** M-26827-1994
**ISSN:** 1134-4792

## EDITORIAL




# EL EFECTO DOMINÓ

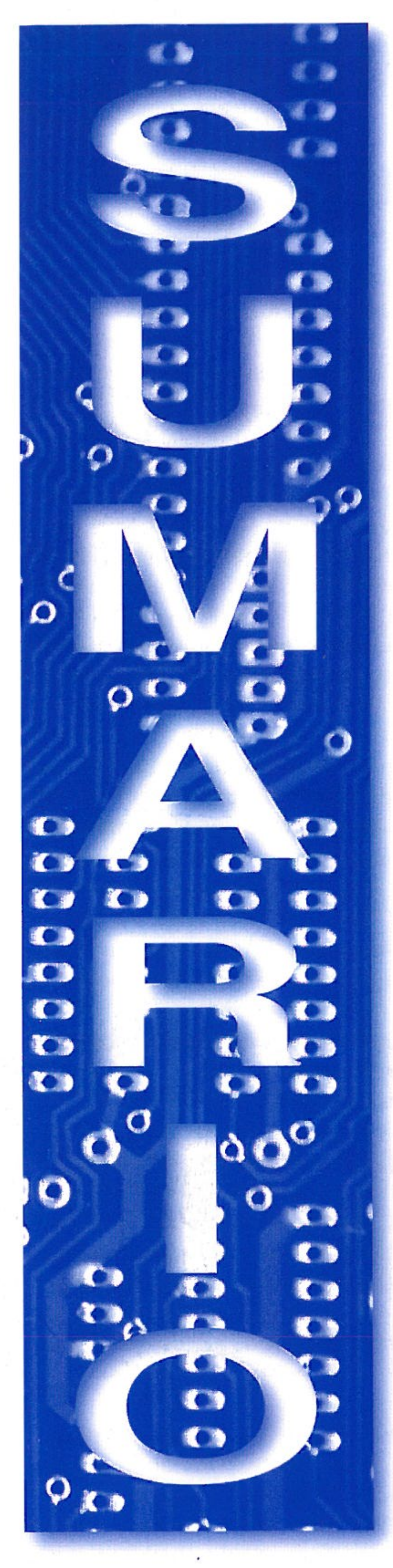
Este mes dedicamos el número de Sólo Programadores al mundo de la Demoscene ya que, además de incluir la habitual sección de nuestro amigo Pedro Antón "Cómo Programar una Demo", incluimos un reportaje sobre la ya tradicional Euskal Party, evento que también Pedro se encargó de cubrir para Sólo Programadores, y que se celebró este pasado verano.

La opinión de los que formamos esta revista exclusivamente dedicada al mundo de la programación es que no hay que tomar estas **parties** como acontecimientos informales donde "esos locos" de la programación acuden y presentan sus trabajos en cuanto se presenta ocasión y pabellón. En realidad el mundo de la programación debe mucho a este tipo de celebraciones, pues el afán de superación de los asistentes produce que en cada Party, o en casi todas, se vean cosas nuevas, como espectaculares efectos y composiciones musicales que dejan en ridículo ciertas melodías que escuchamos diariamente en la radio.

En realidad se podría decir que se produce un efecto dominó, ya que muchas empresas desarrolladoras, especialmente las de vídeo-juegos, asisten con el objetivo de tender sus redes con el propósito de captar los posibles talentos para sus equipos de programación. Estas empresas desarrolladoras de vídeo-juegos invierten dinero y horas en conseguir que sus programas lleven los mejores "engines". Un ejemplo: los juegos tipo Doom / Wolf, ¿hubiera avanzado de semejante manera las técnicas de programación 3D si no hubiera 1000 empresas intentando conseguir el mejor engine 3D posible?. Yo creo que no.

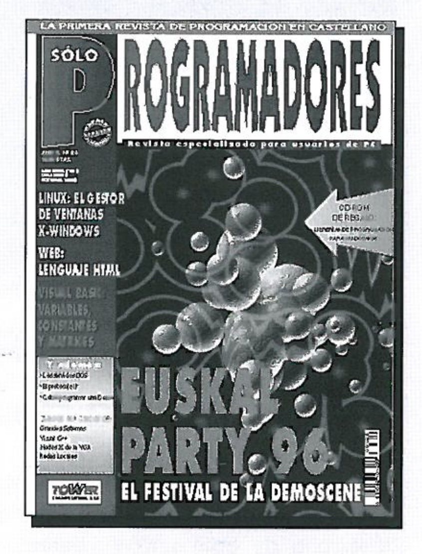
Sigamos con el efecto dominó, alguien exclamará que los juegos no sirven para nada útil. Bueno, eso es muy discutible. Puestos a buscar, podríamos encontrar opiniones médicas respaldando ambas posturas, pero la realidad es que las técnicas empleadas para los vídeo juegos, que no olvidemos que fuera de España es un negocio bastante rentable, pueden ser luego trasladadas a otros campos como son la Medicina, Arquitectura e Ingeniería. Está claro que un equipo de realidad virtual no sólo vale para jugar en entornos 3D, sino que en breve lo veremos incluso sustituyendo, por ejemplo, a los pisos pilotos. Uno podrá pasearse por su futura casa cuando está no tenga ni los cimientos en construcción. Esto es tan sólo un ejemplo, ya que los programadores son los que van a cambiar el mundo en los próximos años. Se puede ser más rotundo: sin programadores no habrá vida.

Por lo tanto, mucho ojo a todos estos "coders" que se presentan en las parties con sus demos a medio hacer, con fallos de polígonos y cuelgues en las rutinas de sonido, porque son los que producen este avance constante en las técnicas de programación. El único "defectillo" de ellos es que todos piensan que su engine 3D es el mejor, y eso que a todos les lleva exactamente 6 meses desarrollarlo.


SUMARIO

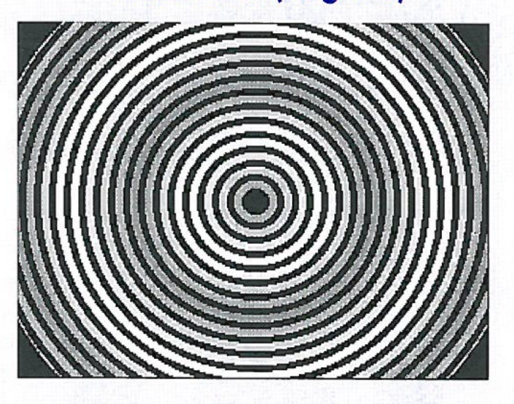

26

## EUSKAL PARTY IV (Pág 39)

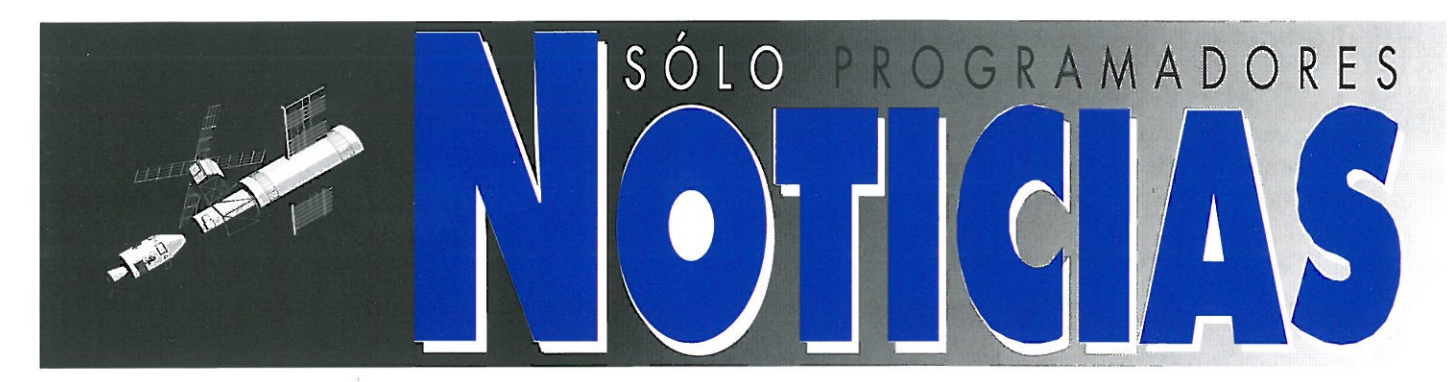



Viene a nuestra portada la demoscene española que se dio cita en San Sebastián en la Euskal, así mismo continua uno de los cursos con más aceptación entre nuestros lectores: Cómo programar una demo, esperamos que algunos de nuestros fieles estén pronto presentando sus trabajos en alguna Party

## MODOS X (Pág. 72)



Este número va dedicado especialmente a los amantes de la programación a bajo nivel, por eso que consideramos de especial interés el artículo de este mes en la sección Modos X de la VGA. También esperamos que sea útil para el lector y dentro de la sección PC Interno el estudio de los servicios DOS




# BORLAND INTRABUILDER

## Nueva familia de productos de desarrollo para intranets

Borland International Inc. ha anunciado la línea de productos IntraBuilder para desarrollo de aplicaciones basadas en Internet en Windows NT y Windows 95. El anuncio fue hecho durante la séptima Conferencia Anual de Desarrolladores, celebrada recientemente.

IntraBuilder, IntraBuilder Profesional e IntraBuilder Cliente/Servidor ofrecen la forma más sencilla para el desarrollo de soluciones de bases de datos basadas en browsers sobre intranets corporativas internas. Combinando las herramientas de bases de datos de Borland con JavaScript, el nuevo lenguaje de desarrollo para Internet, IntraBuilder es una completa suite integrada de herramientas que permiten al programador crear y desplegar sus aplicaciones de forma sencilla.

IntraBuilder soporta cualquier servidor web que corra bajo Windows NT y Windows 95 que soporte el Netscape´s Server Plug-In API o Microsoft ISAPI o CGI. La nueva Herramienta permite el acceso a una gran variedad de formatos de bases de datos, desde Microsoft Access hasta Oracle, incluyendo soporte nativo de Paradox y Visual DBASE. Así mismo, los servidores de bases de datos soportados por los drivers nativos de IntraBuilder incluyen Borland InterBase, Microsoft SQL Server, Oracle, Sybase, Informix e IBM DB2, entre otros. Toda la información sobre este producto, así como una versión de evaluación, están disponibles en la dirección web de Borland *http://www.borland.com/intrabuilder.*

# INNERWEB PUBLISHER Y NETWARE WEBSERVER 2.5

## Publicación de documentos Web para redes corporativas

Novell Inc. acaba de lanzar al mercado Novell InnerWeb Publisher y Netware Web Server 2.5. Además, los usuarios podrán probar Netware Web Server durante 45 días. Estos nuevos productos facilitarán a los clientes publicar, acceder y administrar información en redes corporativas y en Internet. Novell InnerWeb Publisher, uno de los productos más completos de su clase en el mercado, proporciona una solución completa para la publicación de documentos Web internos en un Web corporativo. Este paquete incluye todo lo necesario para hacer de una red una potente plataforma para la publicación de páginas HTML. El producto incluye uno de los servidores Web más rápidos del mercado, un navegador, una herramienta para la creación de páginas Web, un traductor IPX-IP y el sistema operativo Netware 4.1 Runtime.

Como pieza clave de InnerWeb Publisher, Netware Web Server 2.5 permitirá a los usuarios utilizar los servidores Netware como núcleo de sus servidores Web. Este módulo cargable de Netare (NLM) es el único servidor Web que permite a lus usuarios navegar por NDS (*Netware Directory Services*) y acceder rápidamente a la información crítica de usuarios y recursos de red, incluyendo nombres de usuario, direcciones de correo electrónico, URLs y listas de ordenadores y aplicaciones.

Para más información: Novell España
Pere Joan del Valle
Tel: (93) 430 47 10 / Fax: (93) 322 28 90
*http://www.novell.com*

# WEB <TOOLKIT>

## Librería de clases C++ para creación de páginas HTML

ObjectSpace ha anunciado Web<Toolkit>, una librería de clases C++ que soporta la creación de páginas HTML usando un conjunto de clases C++ representando elementos HTML, incluyendo texto, enlaces, gráficos, tablas, formularios y frames. Adicionalmente, esta herramienta proporciona codificación automática de caracteres especiales de texto en la salida HTML y decodificación automática del contenido de los campos en los formularios para ejecutables CGI-bin usando la librería ANSI/ISO Standard Template Library. Web<Toolkit> estará disponible para Pcs al precio aproximado de 45.000 pesetas.

Para más información:
*http://www.objectspace.com*

# VISUAL J++

## Microsoft anuncia la versión beta de su compilador Java

Ya se encuentra disponible en Internet la versión beta de la más esperada herramienta de programación Java, el nuevo Visual J++ de Microsoft. Basado principalmente en un subconjunto de funciones C++, Java proporciona un rápido desarrollo y una amplia portabilidad de las aplicicones. Visual J++ está diseñado para ofrecer a los desarrolladores, tanto noveles como expertos, todo lo necesario para aumentar la rapidez y productividad de la creación de aplicaciones que aprovechen todas las capacidades de Java. Este nuevo paquete incluye el conocido entorno de desarrollo de Microsoft, un debugger gráfico, editor visual, un rápido compilador de código fuente, un completo tutorial *on-line*, orientación a objetos y una larga lista de herramientas para crear potentes páginas y aplicaciones web de una forma rápida y sencilla. Asímismo, Visual J++ es la primera herramienta de desarrollo que permite a los programadores extender sus aplicaciones utilizando componentes creados con otros lenguajes de programación que soporten la especificación ActiveX. De esta forma, Microsoft define un nuevo estándar Java en Visual J++, como ya ocurriera anteriormente con C y C++. Tanto la beta de Visual J++ como toda la información acerca de este producto están disponibles en el web de Microsoft en la dirección *http://www.microsoft.com/visualj.*

# VISUAL COMPONENT LIBRARY

## Librería de componentes Delphi para Crystal Reports

TransactNet ha anunciado recientemente Web Interface Toolkit (WIT), una nueva herramienta que permite a los desarrolladores construir clases Java para acceder a los datos y servicios Web de forma rápida y sencilla desde aplicaciones servidor. Combinando el análisis HTTP con el soporte de protocolos HTTP y CGI, WIT abre la puesta en escena de la próxima fase en el desarrollo de aplicaciones para Internet e intranets: el acceso automático a los documentos y servicios Web. Al incluir un set de librerías de clases Java, generadores de código y un intuitivo interfaz de usuario, WIT puede generar applets Java total o parcialmente y aplicaciones independientes. Web Interface Toolkit funciona en todas las plataformas que soporten el Java Development Kit (JDK), incluyendo Windows 95/NT, Solaris, AIX, Linux y Macintosh. Los programas creados con WIT podrán comunicarse con aplicaciones Web escritas en otros lenguajes como Java, C,C++,Perl,Tcl y otros... Se encuentra disponible una beta gratuita de WIT 1.0 en la dirección *http://www.transactnet.com.*

# INFERNO

## Nuevo sistema operativo desarrollado en Bell Labs

Lucent Technologies, antes conocida como Bell Labs, ha anunciado Inferno, un sistema operativo que soporta aplicaciones altamente interactivas para prácticamente cualquier ordenador o sistema de entretenimiento sobre cualquier red de comunicaciones, desde la telefonía y la televisión por cable a la comunicación vía satélite e internet. Los requerimientos de Inferno son mínimos y usará menos de 1 Mb de memoria RAM.

La oferta de Inferno incluye el sistema operativo de red, los protocolos de comunicaciones (Styx), el entorno de programación de red, el lenguaje de desarrollo (Limbo), y la máquinq virtual (Dis). Inferno correrá como un sistema operativo independiente en máquinas con procesadores Intel x86, Mips y ARM, y podrá ser situado como entorno virtual en sistemas UNIX y Windows 95/NT.

Inferno ha sido desarrollado por Rob Pike, Phill Winterbottom y Sean Dorward, del Centro de Desarrollo de Bell Labs, donde se crearon Unix y C/C++. Para más información sobre inferno consultar en Internet la dirección *http://inferno.bell-labs.com/inferno/*

# INTERNET PHONE

## Intel enriquece las comunicaciones "persona-a-persona" en Internet

Intel ha anunciado recientemente una nueva aplicación de telefonía en internet para plataformas basadas en Windows 95 diseñada para interoperar con un amplio abanico de software de comunicaciones basados en H.323. Este nuevo producto incorpora la tecnología *User Listing Service* (*ULS*) de Microsoft para localizar de forma sencilla a otros usuarios de productos software de telefonía mediante los servicios de guías ya existentes en Internet. Una vez conectados, los usuarios podrán desde su PC multimedia conversar en Internet utilizando a la vez otras aplicaciones que permitirán, por ejemplo, visitar el Web con su interlocutor, intercambiar imágenes, datos, etc... Asímismo, existe una versión beta del producto en el Web de Intel para todo aquel que quiera probar el procuto, así como información más detallada sobre él en la dirección *http://www.intel.com.iaweb.cpc*

# PUENTE ENTRE JAVA Y C++
## Nueva tecnología Java de ILOG y SunSoft

ILOG Inc. y SunSoft han anunciado un acuerdo para desarrollar un puente entre Java y C++. El proyecto, conocido como "TwinPeaks", cimentará el camino para conseguir una tecnología que soporte la totalidad de los componentes estándar C++ y ANSI C en Java. Este puente será un programa "pasarela" que permitirá a Java usar y reconocer los componentes C++ a través de un interfaz generado automáticamente.

Los componentes C++ están formados por dos elementos: un interfaz o fichero de cabecera y el código objeto. El "puente TwinPeaks" podrá leer y analizar estos ficheros de cabecera y creará todo un interfaz Java y un pequeño fragmento de código Java/C++ que convertirá las llamadas API y los formatos de datos de los componentes existentes. A través de esta conversión, Java estará capacitado para reconocer y usar los componentes C++ existentes.

# IBM DB/2
## Orientación a Objetos para DB/2

IBM continúa dotando a su base de datos relacional DB/2 de mayores contenidos multimedia y tecnología orientada a objetos, después de dar a conocer las nuevas extensiones multimedia que han sido incluidas en una de las bases de datos de más prestigio del mercado. Estas extensiones permitirán gestionar, manípular y distribuir eficazmente complejos datos multimedia compuestos de textos, imágenes, vídeo y audio en lo que representa la culminación de una estrategia dirigida a dotar de características de orientación a objetos de DB/2.

Disponibles para servidores y clientes basados en las plataformas operativas AIX, OS/2 y Windows NT, estas soluciones constituyen el componente clave de aproximación a la tecnología orientada a objetos de IBM en bases de datos, una iniciativa muy relacionada con las propuestas de sus principales competidores para lanzar servidores basados en tecnología orientada a objetos.

Por otro lado, la firma también anunció la disponibilidad de la versión 1.2 de DB/2 Parallel Edition, que ofrece a los usuarios una mejora aproximada del 50 por ciento en la relación de rendimientos de *queries* y de un 200 por ciento en la velocidad del proceso de transacciones.

# JUST-IN-TIME
## Compilador para Java de Symantec

Symantec ha anunciado la disponibilidad de su compilador Just-In-Time para Java, de forma gratuita, para los usuarios de Café. Este compilador transforma los *bytescodes* de Java en instrucciones de procesador nativo, para lograr una mayor rapidez de ejecución bajo 32 bits.

El resultado de este proceso es una mejora de hasta 23 veces superior en la velocidad de ejecución de los *applets* Java y aplicaciones estándar. Al proporcionar Just-In-Time, Symantec posibilita una ejecución con velocidad similar al código nativo C/C++ para los *applets* web y las aplicaciones estándar de Java. De esta forma, Symantec toma parte en la carrera por obtener la estandarización de Java, como ya han hecho también Microsoft, Borland y otras muchas empresas.

El nuevo compilador de Symantec está disponible en la dirección web de la compañía para todos los compradores iniciales de Symantec Café.

Para más información:
*www.cafe.symentec.com*

# WEB INTERFACE TOOLKIT
## Nueva herramienta de automatización de Web de TransactNet

TransactNet ha anunciado recientemente Web Interface Toolkit (WIT), una nueva herramienta que permite a los desarrolladores construir clases Java para acceder a los datos y servicios Web de forma rápida y sencilla desde aplicaciones servidor. Combinando el análisis HTTP con el soporte de protocolos HTTP y CGI, WIT abre la puesta en escena de la próxima fase en el desarrollo de aplicaciones para Internet e intranets: el acceso automático a los documentos y servicios Web. Al incluir un set de librerías de clases Java, generadores de código y un intuitivo interfaz de usuario, WIT puede generar applets Java total o parcialmente y aplicaciones independientes. Web Interface Toolkit funciona en todas las plataformas que soporten el Java Development Kit (JDK), incluyendo Windows 95/NT, Solaris, AIX, Linux y Macintosh. Los programas creados con WIT podrán comunicarse con aplicaciones Web escritas en otros lenguajes como Java, C,C++,Perl,Tcl y otros... Se encuentra disponible una beta gratuita de WIT 1.0 en la dirección *http://www.transactnet.com.*

# WORLD WIDE WEB
## Demanda de empleo en Internet

Desde hace tiempo, las empresas pueden consultar a través de Internet una serie de curriculums de estudiantes de la Facultad de Informática y de la Escuela Universitaria de Informática de la UPM.

Con este proyecto se pretende que las empresas puedan acceder rápidamente a los curriculums de estudiantes de Ingeniería Informática, que necesitan la experiencia laboral como parte de su aprendizaje, y que pueden aportar a las empresas el resultado de una formación tan global y específica como la recibida en la Universidad. Cada curriculum se identifica por una dirección de correo en la que las empresas pueden dejar los mensajes para aquellos alumnos que se ajusten al perfil de la vacante. Además, en algunos de ellos aparece un enlace con la página personal de ese alumno que puede contener el curriculum ampliado, fotografía, etc... Además, algunos de los alumnos añaden su número de teléfono u otras formas de contacto. El Web esta ubicado en el servidor de alumnos de la Facultad de Informática, en la cuenta personal del promotor de la idea, cuya dirección es *http://zipi.fi.upm.es/%7ea890248/*

# BROKER 2.1
## ILOG aumenta la potencia de C++

ILOG ha anunciado recientemente la integración de Broker 2.1 con Orbix, una implementación de CORBA (Common Object Request Broker Architecture) de IONA Technologies, que permitirá a los programadores en C++ desarrollar fácilmente aplicaciones usando únicamente las clases C++ nativas para sus aplicaciones. CORBA es la vía estándar de obtener aplicaciones interoperativas en varias plataformas y lenguajes. Esta nueva implementación de CORBA soporta 20 tipos distintos de plataformas y trabaja con un amplio número de lenguajes, incluyendo C++, Java, Ada y Visual Basic. De esta forma, Broker 2.1 aumenta la potencia de C++ con una serie de palabras clave con las que los programadores podrán anotar cabeceras para exportar clases, plantillas y funciones. Broker usará estas cabeceras para generar interfaces IDL asociados y fragmentos de implementación. Toda la semántica C++ está totalmente soportada, incluyendo objetos, funciones virtuales y estáticas, sobrecarga de funciones y objetos transitorios. ILOG Broker 2.1 soportará plataformas UNIX, Windows NT y Windows 95.

Para más información:
*http://www.ilog.com*

# NETIMPACT STUDIO
## Desarrollo de aplicaciones profesionales en WWW

Durante su Quinta Conferencia Anual de Usuarios, Powersoft anunció NetImpact Studio, un entorno intuitivo para desarrollo de aplicaciones profesionales en World Wide Web. Esto posibilita el desarrollo de aplicaciones Web de envío de datos dinámicas usando HTML, JavaScript, componentes y SQL o aplicaciones servidor operativas. NetImpact Studio incluye todas las herramientas y servicios necesarios para desarrollar, probar y desplegar aplicaciones profesionales para Internet e intranets corporativas. Asímismo ofrece un sofisticado interfaz gráfico, conectividad de bases de datos, depurador de código, entorno ensamblador y otras funcionalidades que facilitan el rápido desarrollo de las aplicaciones. NetImpact Studio soportará una amplia gama de servidores Web y APIs, incluyendo Netscape's NSAPI, Microsoft ISAPI y CGI y permitirá el desarrollo en Windows 95 y Windows NT.

Para más información:
*http://www.powersoft.com*

# CAFÉ 1.5
## Actualización del entorno de desarrollo de Symantec

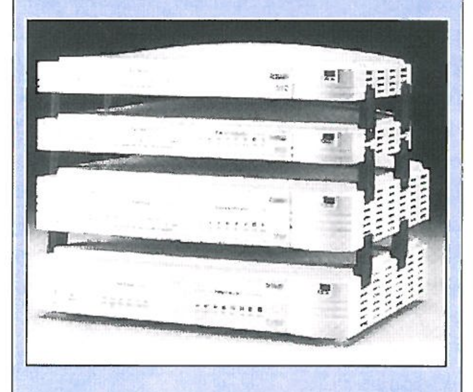
Symantec Corporation ha anunciado la última actualización de su popular herramienta de desarrollo Java Symantec Café. Café es una herramienta de programación Java visual que incluye el conocido entorno integrado de desarrollo de Symantec. Las nuevas funcionalidades de Symantec Café incluyen un nuevo compilador que es cinco veces más rápido que el anterior y nuevas mejoras en el depurador, incluyendo la evaluación de expresiones, con el objetivo de proporcionar un mayor control sobre las aplicaciones y reducir el tiempo de depuración. Symantec ha anunciado que la nueva actualización soportará el Netscape ONE (Open Network Environment) al incluir las nuevas Netscape Internet Foundation Classes en futuras versiones de Café. El compilador funcionará en plataformas Windows 95 y NT, y estará también disponible para entornos Macintosh y Power Macintosh.

Para más información:
*http://www.symantec.com*

# NOVEDADES

## Soluciones de conectividad de 3 COM

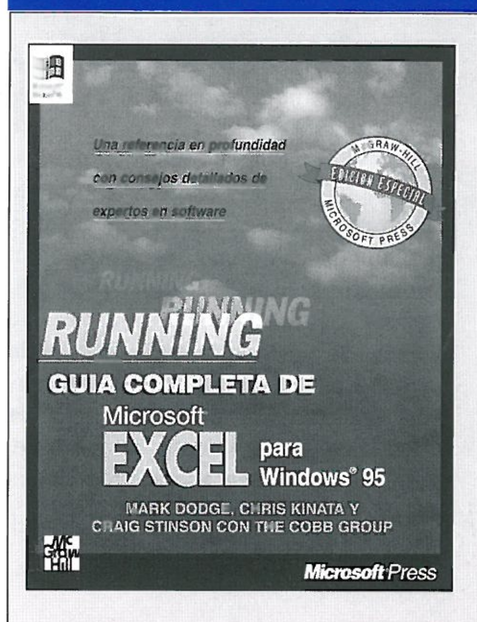
3Com acaba de lanzar la familia de productos OfficeConnect. Esta nueva línea de módulos de interconexión permite a las empresas mejorar la productividad interconectando sus recursos. La elección de estos tres Hubs OfficeConnect proporciona conexión central para ocho ordenadores. Cada uno de estos hubs permite alta velocidad en la conexión de los recursos. Los servidores de impresión y fax OfficeConnect proporcionan, además, el reparto de los servicios de impresión y fax de red, respectivamente. La nueva familia de productos de 3Com proporciona todas las capacidades necesarias para compartir todos los recursos de forma efectiva.



Las principales ventajas de estos módulos radican en su velocidad de instalación, facilidad de uso y administración y su diseño, pensado para ocupar el minimo espacio. La línea OfficeConnect está pensada para facilitar su rápida instalación. Para instalar un hub OfficeConnect tan sólo hay que enchufar un cable trenzado desde cada PC hacia cada uno de los ocho puertos RJ-45 del propio hub, y encenderlo. La configuración es tan simple como le de Windows 95 u otros productos de conectividad.

Una vez en funcionamiento, OfficeConnect requiere una mínima atención. Dependiendo del modelo, varios paneles frontales proporcionan una rápida información del estado de la red. Por ejemplo, todos los modelos OfficeConnect incluyen un LED de alerta que permite al usuario menos experimentado identificar rápidamente los problemas de la red.

# LIBROS



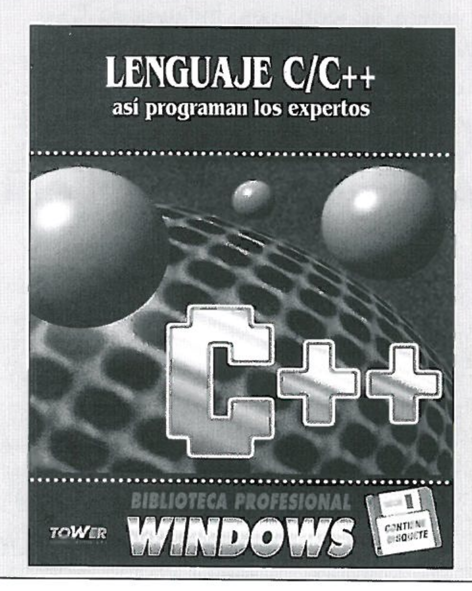

## Running, guía completa de Microsoft Excel para Windows 95

Este libro es una completa guía de usuario y consulta, además de una fuente inapreciable de información detallada y precisa para aprovechar al máximo las nuevas posibilidades ofrecidas por Excel 7.0 para Windows 95. Contiene instrucciones paso a paso fáciles de seguir, ejemplos realistas, ilustraciones de pantalla y consejos de expertos que le mostrarán cómo utilizar esta herramienta del modo más eficiente posible. El libro está orientado a usuarios nuevos y también a los ya iniciados para que lleguen a ser verdaderos expertos. La guía completa de Microsoft Excel está estructurada en los siguientes temas: Fundamentos de Hojas de Cálculo, Cómo trabajar de una forma eficiente, Ánalisis de datos, Diagramas y gráficos innovadores, Gestión de información, Trabajo con macros y la utilización de una aplicación de ejemplo creada con Visual Basic para resolver el problema común.

Editorial: Mc Graw Hill
Autor: Mark Dodge
1200 páginas
Idioma: Castellano

## Lenguaje C/C++ así programan los expertos

El lenguaje C se ha empleado para desarrollar además de incontables utilidades, aplicaciones y herramientas de programación, el propio sistema operativo DOS, el Unix y todo el entorno Windows. De hecho, el C++, siendo una ampliación del lenguaje C original, debe su exixtencia, entre otros, a las exigencias de la programación windows. La intención fundamental de este libro es, por tanto, introducir al lector en el mundo de la programación en C, y despejar todas esas dudas y temores iniciales a introducirse en el mundo del código.




Autor: Javier Guadalajara
Editorial: Tower Communications
190 páginas
idioma: Castellano

# ¡SORTEO DE 50 CONEXIONES A INTERNET!

## RELLENA ESTA ENCUESTA Y PARTICIPA EN EL SORTEO

**SECCIONES** **VALÓRALAS DE 1 A 10**

| SECCIONES | VALÓRALAS DE 1 A 10 |
|---|---|
| NOTICIAS Y LIBROS | 1 2 3 4 5 6 7 8 9 10 |
| CARTAS DEL ILUSO | 1 2 3 4 5 6 7 8 9 10 |
| TECNOLOGÍA WEB | 1 2 3 4 5 6 7 8 9 10 |
| CURSO DE PROGRAMACIÓN | 1 2 3 4 5 6 7 8 9 10 |
| SISTEMAS ABIERTOS | 1 2 3 4 5 6 7 8 9 10 |
| GRANDES SISTEMAS | 1 2 3 4 5 6 7 8 9 10 |
| CURSO DE REDES | 1 2 3 4 5 6 7 8 9 10 |
| SIST. OPERATIVO LINUX | 1 2 3 4 5 6 7 8 9 10 |
| CÓMO PROG. UNA DEMO | 1 2 3 4 5 6 7 8 9 10 |
| MODOS X DE LA VGA | 1 2 3 4 5 6 7 8 9 10 |
| PC INTERNO | 1 2 3 4 5 6 7 8 9 10 |
| CURSO VISUAL BASIC | 1 2 3 4 5 6 7 8 9 10 |
| CURSO VISUAL C++ | 1 2 3 4 5 6 7 8 9 10 |

¿Qué te parece el nivel de los artículos?

- ☐ Muy alto ☐ Alto
- ☐ Normal ☐ Bajo
- ☐ ..................................

¿Qué te parece en general el contenido de la revista?

- ☐ Muy bueno ☐ Bueno
- ☐ Regular ☐ Malo
- ☐ ..................................

¿Qué cambios te gustaría realizar? ¿Cúales?

- ☐ AÑADIR NUEVAS SECCIONES ________ ________ ________
- ☐ QUITAR SECCIONES ________ ________ ________
- ☐ AMPLIAR LAS ACTUALES ________ ________ ________

¿Te conectas a algún servicio On-Line?

- ☐ Sí ☐ No
- ☐ Habitualmente ☐ Esporádicamente

¿Cúal?

- ☐ INTERNET
- ☐ INFOVÍA
- ☐ COMPUSERVE
- ☐ BBS
- ☐ OTROS

**DATOS PERSONALES**

Nombre y Apellidos ____________________

Domicilio ____________________

Población ____________________

C.P. ____________________

Provincia ____________________

Teléfono ____________________

Edad ____________________

Estudios ____________________

Profesión ____________________

Ordenador de que dispone ____________________

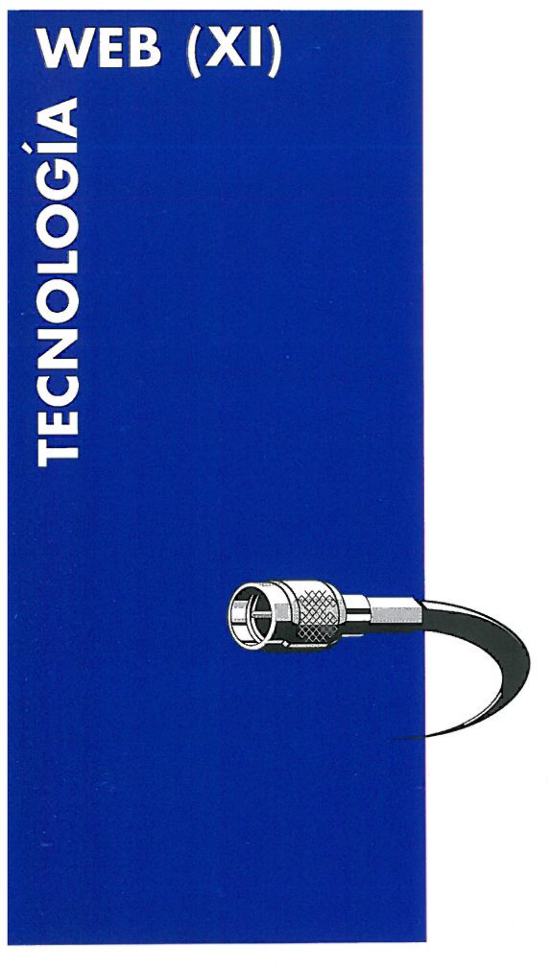
Configuración de tu equipo:

| | | | | | |
|---|---|---|---|---|---|
| Procesador | 386 | 486 | PENTIUM < 120 | PENTIUM > 120 | |
| Tarjeta Gráfica | VGA | SVGA | 65K COLORES | 16,7M COLORES | |
| Tarjeta de Sonido | SOUND BLASTER | SOUND BLASTER 16 | GRAVIS ULTRA | | |
| CD-ROM | 1X | 2X | 4X | 6X | 8X |
| Módem | 2.400 | 9.600 | 14.400 | 28.800 | |

**RELLENA EL CUESTIONARIO Y ENVÍALO A:**

**SÓLO PROGRAMADORES (Tower Communications)**
**Sorteo Internet**
**C/ Aragoneses, 7 - 28100 Pol. Ind. Alcobendas (Madrid)**

**Tu revista favorita y la empresa Datagrama te ofrecen la posibilidad de participar en un sorteo de 50 cuentas de conexión a Internet. Para participar, sólo tienes que rellenar los datos que aparecen en esta página y enviarla (sirve fotocopia) a la dirección que aparece unas líneas más arriba.**
**Nota: Las conexiones son efectivas durante un mes sin limitación diaria.**



# LENGUAJE HTML (V):

## Imágenes con mapa en el cliente y otras extensiones

*Fernando J. Echevarrieta*

**Con el presente artículo se introducen nuevas extensiones que proporcionarán un aspecto más artístico y profesional a las páginas de WWW, desde la incorporación de imágenes con mapa en el cliente, lo que las hace por fin accesibles a todos los usuarios, hasta los forms para el envío de ficheros, pasando por nuevas etiquetas dedicadas a una mejora global de la presentación**

Un mes más la sección se mueve por el mundo del HTML. Mundo que, lejos de agotarse, continúa creciendo como un ser vivo. En especial cabe destacar el abandono del W3 Consortium de sus esperanzas por estandarizar una versión 3.0 de HTML que se había convertido en algo demasiado complejo e inmanejable, para dar lugar a la actual versión 3.2, formada principalmente por los estándares de facto introducidos por Netscape. Próximamente se dedicará un artículo a este tema. Por el momento, se expondrá la mayoría de las extensiones que se han ido proponiendo mientras se trataba de estandarizar el HTML 3.0 (recuérdese que "un camello es un caballo realizado por un comité"), como la incorporación de imágenes con mapa en el cliente (según la propuesta de Spyglass), la posibilidad de envío de ficheros a un servidor desde forms (siguiendo la propuesta de Xerox Parc) y los nuevos atributos para la justificación de textos en forms y las etiquetas para la mejora de la presentación e inclusión de *plug-ins* propuestas por Netscape.

### SERVER-SIDE VS CLIENT-SIDE IMAGEMAPS

En el tercer artículo de la serie, el segundo dedicado a HTML [Echeva-1], se trataba una de las características más vistosas, hasta ese momento, de la telaraña: el empleo de imágenes mapa, o imágenes con zonas sensibles que conducían a uno u otro destino en función de sobre qué zona caliente se pulsara.

Originalmente, el funcionamiento de los mapas de imágenes se encontraba basado en la existencia de un CGI en el servidor. De este modo, al pulsar el usuario sobre una imagen, el browser de WWW enviaba las coordenadas de la pulsación a este servidor, donde se lanzaba el CGI que se encargaba de realizar una redirección al lugar adecuado tras consultar el mapa preparado al efecto. Es por esto que el mecanismo tradicional ha dado en llamarse *server-side imagemaps* (imágenes mapa en el lado del servidor).

Si bien el diseñador de páginas HTML no tenía necesidad de saber nada acerca de CGI, sino que, simplemente, debía conocer la forma de definir la geometría de los mapas, este mecanismo implicaba una serie de limitaciones:

En primer lugar, se encontraba diseñado para funcionar únicamente sobre HTTP, por lo que no era posible utilizarlo sobre otros protocolos (lo que, por el momento, sigue sin ser un problema), ni de forma local. Este último dato ha sido la herida más sangrante del sistema convencional ya que, desde el punto de vista del desarrollador, no era posible implementar páginas con mapas sensibles y probarlas si no se contaba con un servidor de HTTP y, desde el punto de vista de los usuarios, no era posible emplear una imagen mapa sin una conexión Internet al servidor apropiado. Por ello, carecía de sentido realizar *mirrors* o copias locales de documentos que contuvieran este tipo de mapas ya que, en local, son inservibles.

Por la misma razón, el uso de imágenes mapa introducía un *overhead* de, al menos, dos conexiones HTTP, al ser necesario enviar un mensaje al servidor y esperar la llegada de la respuesta

| TABLA 1 | |
|---|---|
| <IMG USEMAP=> | La imagen tiene un mapa en un documento HTML |
| <MAP [NAME=]>.. </MAP> | Límites y nombre del mapa |
| <AREA [SHAPE=RECTIPOLYICIRC] COORDS= HREFINOHREF> | Geometría de las zonas calientes |

**Etiquetas para imágenes con mapa en el cliente.**

consecuente para poder realizar una nueva conexión. Este problema se veía acrecentado en servidores de difícil acceso (bajo ancho de banda, alta latencia) o en los casos en que se pulsaba sobre una zona que no debía conducir a ninguna parte.

Otro de los puntos negros, derivado a su vez de los anteriores es que, al ser necesaria la intervención de un CGI, con el sistema tradicional no es posible conocer a priori a dónde conducirá una pulsación. Esto es posible con el resto de los enlaces, ya que al pasar sobre ellos el ratón la mayoría de los browsers muestran el URL destino (si el autor del documento no se ha apresurado a copiarse y hacer "abuso" de aquel famoso programa en *javascript* que se encarga de mostrar un texto en scroll horizontal precisamente donde deben mostrarse los URLs) y, en el peor de los casos, siempre es posible acceder a la fuente HTML del documento para ver a dónde conducen.

Por último, al tratarse de un mecanismo dependiente de un CGI, se limita en gran medida la portabilidad de un servidor a otro. Así, se hace necesario, bien instalar la correspondiente versión del CGI en el nuevo servidor, bien modificar los ficheros mapa para adaptarse a la sintaxis conocida por el nuevo CGI [Echeva-4].

Por todo ello, desde el primer momento se intuyó la necesidad de trasladar la función de análisis de las coordenadas al cliente y, a ser posible, de permitir incorporar la geometría de los mapas en los propios documentos HTML. Este mecanismo es el que adoptaría el nombre de *client-side imagemaps* (imágenes mapa en el lado del cliente). La primera propuesta de un mecanismo *client-side* partió de Spyglass, que lo incorporó a sus browsers Mosaic. Sorprendentemente, y dado que la propuesta era, por su simplicidad, tan buena como cualquier otra, es la que han ido adoptando otros fabricantes como Netscape. Hoy en día, los *server-side imagemaps* son cada vez menos empleados en Internet.

## IMÁGENES CON MAPA EN EL CLIENTE

Con el fin de hacer posible la incorporación de *client-side imagemaps* a documentos HTML, se amplía la etiqueta *<IMG>* mediante la adición de un nuevo atributo: *USEMAP*, que califica a la imagen en cuestión como imagen-mapa, de la misma forma que *ISMAP* lo hacía con los *server-side imagemaps* [Echeva-1]. En aquel caso era necesario hacer de la imagen un enlace al CGI encargado de procesar las coordenadas, al cual se le indicaba el fichero de definición de la geometría de las zonas calientes mediante una información de path adicional [Echeva-2]. Ahora, el mecanismo es más simple, ya que no es necesario invocar un CGI, pero tanto si se utiliza el método clásico como con los mapas en el lado del cliente, no es suficiente con especificar qué imagen actuará como mapa sino que también será necesario especificar la geometría de las zonas calientes mediante un mapa lógico.

En el caso de imágenes mapa en el lado del cliente, el mapa lógico se encuentra incrustado en un documento HTML y se indica su localización mediante el valor del atributo *USEMAP*. *USEMAP* puede tomar como valor cualquier nombre de "ancla" en un documento HTML, es decir, cualquier URL seguido del signo *hash* (#) y el nombre de un mapa. Si no se especifica URL, se entiende que el mapa se encuentra incrustado en el mismo documento HTML. Así pues, una posible imagen mapa se especificaría de la forma:

*<IMG SRC="imagen.gif" USEMAP=#mapa1>*

que equivaldría a lo que hasta ahora se indicaba como:

*<A HREF=/cgi-bin/htimage/mapa1>*
*<IMG SRC="imagen.gif" ISMAP>*
*</A>*

En principio, la idea de emplear un mapa que se encuentra en otro documento podría parecer un paso atrás ya que, de nuevo, será necesario realizar una conexión con el servidor para recuperar el mapa. Sin embargo, puede resultar de gran utilidad a la hora de realizar el diseño y mantenimiento del servidor ya que varias documentos podrán compartir el mismo mapa y, realizando un sólo cambio, variará el comportamiento de todos. Este caso es frecuente en servidores multilingües. También se suele emplear cuando existen pocos mapas y éstos son sencillos, alojándolos todos en un solo documento con el fin de mantenerlos centralizados. Así, por ejemplo, el uso de un mapa contenido en otro documento se podría indicar como:

*<IMG SRC=imagen.gif USEMAP="/mapas.html#mapa1">*

Se recuerda al lector que esta notación ya se empleaba en HTML 2.0 para apuntar a lugares concretos de un documento que se señalizaban mediante el atributo *NAME* de la etiqueta *<A>*, por ejemplo *<A NAME="punto1">* [Echeva-2].

## MANTENIENDO LA COMPATIBILIDAD

Como habrá advertido ya el lector, a través de su andadura por los artículos de esta sección, el lenguaje HTML es algo vivo. Diversos grupos de trabajo realizan nuevas propuestas que se incorporan a nuevos browsers. Si la propuesta es aceptada y tiene sentido, se adopta. Si no, se descarta siguiendo un proceso de "selección natural". Pero siempre, como norma, se debe ser compatible con el resto de los browsers.

Los *server-side imagemaps* no iban a ser una excepción a esta regla. Así, es posible escribir código HTML que pueda ser interpretado correctamente tanto por un browser que incorpore la capacidad de analizar mapas en el cliente como por uno que no lo haga. Volviendo sobre el mismo ejemplo:

*<A HREF="/cgi-bin/htimage/mapa1">*
*<IMG SRC="imagen.gif" USEMAP="#mapa1" ISMAP>*
*</A>*

funcionará en cualquier browser. Aquel que entienda la etiqueta *USEMAP* sabrá que se trata de una imagen con mapa en el cliente y al pulsar sobre ella la tratará como tal aunque se encuentre en un enlace. Si, por el contrario, la etiqueta *USEMAP* no es reconocida, será ignorada, con lo que el código interpretado será el habitual de una imagen con mapa en el servidor.

Aun si se ha decidido no dar soporte (o no es posible) a mapas en el servidor, es interesante utilizar esta última característica para dirigir al usuario a una página alternativa, por ejemplo, con el mismo menú en forma textual:

*<A HREF=URL_alternativa>*
*<IMG SRC="imagen.gif"*
*USEMAP="#mapa1">*
*</A>*

## ESPECIFICANDO LA GEOMETRÍA DE UN MAPA

Hasta el momento, únicamente se ha indicado cómo apuntar a un mapa lógico pero, obviamente, será necesario haberlo definido previamente. Con este fin, se añaden dos nuevas etiquetas al lenguaje *<MAP>...</MAP>* y *<AREA>*:

- ***<MAP>...</MAP>***: tiene como único objetivo fijar los límites de definición de las zonas calientes de un mapa y darle nombre a través de su único atributo *NAME*.
- ***<AREA>***: Es la etiqueta destinada a definir las zonas calientes y a dónde deben apuntar. Para ello, emplea tres parámetros:

*SHAPE*, que especifica la forma geométrica de la zona caliente y puede tomar los valores : *RECT*, *POLYGON* y *CIRCLE*. En caso de no especificarse este atributo se considera *RECT* como valor por defecto

*COORDS*, que indica las coordenadas que definen la forma geométrica especificada y

*HREF*, que indica el URL del objeto apuntado.

Las coordenadas correspondientes a cada figura son de la forma:

*<AREA SHAPE=RECT*
*COORDS=x1,y1,x2,y2 HREF=URL>*

donde *x1,y1* son las coordenadas del vértice superior izquierdo y *x2,y2* las del inferior derecho

*<AREA SHAPE=POLYGON*
*COORDS=x1,y1,x2,y2,...xn,yn*
*HREF=URL>*

donde *xi,xi* son las coordenadas de cada uno de los vértices del polígono

*<AREA SHAPE=CIRCLE COORDS=x,y,r*
*HREF=URL>*

donde *x,y* son las coordenadas del centro y *r*, el radio.

Así pues, el mapa lógico asociado a la figura 1, que se empleó como ejemplo en [Echeva-1] se definiría como:

*<MAP NAME="SoloP">*
*<AREA SHAPE=RECT*
*COORDS=38,46,173,150*
*HREF=http://hohoo/ncsa.uiuc.edu/docs/tutorials/imagemapping.html>*
*<AREA SHAPE=RECT*
*COORDS=223,47,361,150*
*HREF=http://www.w3.org/hypertext/WWW/Daemon/User/CGI/HTImageDoc.html>*
*<AREA SHAPE=CIRCLE*
*COORDS=200,14,52*
*HREF=http://highland.dit.upm.es:8000>*
*</MAP>*

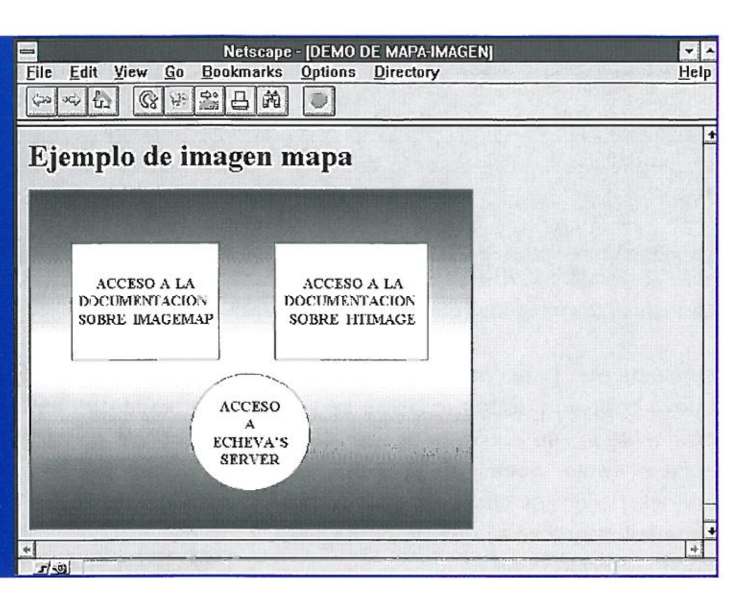



**Figura 1: Ejemplo que en su día se empleó para ilustrar los *server-side imagemaps*, cuyas definiciones se pueden observar en el listado 1 y que, en el presente artículo se define como *client-sede*.**

**LISTADO 1**

**Definición con imagemap**

```
default http://highland.dit.upm.es:8000/sp/
fotomapa.html
rect http://hoohoo.ncsa.uiuc.edu/docs/tutorials/
imagemapping.html 38,46 173,150
rect http://www.w3.org/hypertext/WWW/
Daemon/User/CGI/HTImageDoc.html 223,47
361,150
circle http://highland.dit.upm.es:8000 200,214
252,214
```

**Definición con htimage**

```
default http://highland.dit.upm.es:8000/sp/
fotomapa.html
rectangle ( 38,46) (173,150)
http://hoohoo.ncsa.uiuc.edu/docs/tutorials/
imagemapping.html
rectangle (223,47) (361,150)
http://www.w3.org/hypertext/WWW/Daemon/
User/CGI/HTImageDoc.html
circle (200,214) 52
http://highland.dit.upm.es:8000
```

**Server-side imagemaps correspondientes al ejemplo del artículo y a la figura 1.**

Compárese con la forma de definir los mapas que se empleaba con los CGIs *htimage* (del CERN) e *imagemap* (de NCSA) y que aparece en el listado 1.

## ZONAS SUPERPUESTAS Y VALORES POR DEFECTO

Como se observará al examinar el listado 1, en la definición de los *client-side imagemaps*, se echa en falta la forma de indicar un valor por defecto, y es que se considera que cualquier zona de la imagen que no haya sido definida como zona caliente debe ser insensible a los *clicks* del ratón. Esto es una ventaja añadida ya que con el método tradicional se solía emplear casi siempre el valor *default* para volver a cargar el mismo documento, con la consiguiente espera por los accesos a red y, si por olvido, no se indicaba un valor *default* y las zonas calientes no cubrían determinada área, al pulsar sobre ésta el CGI producía un error (nuevamente, tras la correspondiente espera).

Al igual que con los mapas en el servidor, aún se conserva la idea de una





**LISTADO 2**

```
<a href="/cgi-bin/imagemap/images/home.map">
<img align="center" width=511 height=454
alt="Andrew Lloyd Webber" src="images/home.gif"
border=0

USEMAP="#home" ISMAP></a>
(...)
<MAP NAME="home">
<AREA SHAPE="rect" HREF="/Joseph/index.html"
COORDS="38,1,104,72">
<AREA SHAPE="rect"
HREF="/Starlight/index.html"
COORDS="133,2,194,73">
<AREA SHAPE="rect" HREF="/productions/
otherprods.html#Aspects"
COORDS="222,1,285,67">
<AREA SHAPE="rect" HREF="/productions/
otherprods.html#Jesus"
COORDS="314,2,376,72">
<AREA SHAPE="rect" HREF="/productions/
otherprods.html#Evita"
COORDS="405,1,471,72">
<AREA SHAPE="rect" HREF="/cgi-bin/rbox/
store.cgi" COORDS="3,386,58,453">
<AREA SHAPE="rect" HREF="/recordings/
index.html" COORDS="76,385,131,453">
<AREA SHAPE="rect"
HREF="/productions/index.cgi"
COORDS="148,386,204,453">
<AREA SHAPE="rect" HREF="/news/index.html"
COORDS="221,386,280,453">
<AREA SHAPE="rect"
HREF="http://206.67.227.6/RUG/bbs/index.html"
COORDS="297,387,357,453">
<AREA SHAPE="rect" HREF="/alw.html"
COORDS="375,387,434,453">
<AREA SHAPE="rect" HREF="/quiz/index.html"
COORDS="452,386,509,453">
<AREA SHAPE="poly" HREF="/recordings/
index.html"
COORDS="208,74,227,117,277,117,295,69,203,69">
<AREA SHAPE="rect" HREF="/reallyuseful.html"
COORDS="110,324,384,381">
<AREA SHAPE="poly" HREF="/Sunset/index.html"
COORDS="78,92,48,131,35,162,33,178,32,211,45,
215,45,243,48,247,47,253,42,252,34,259,34,347,16
0,276,155,260,160,245,174,243,192,242,211,245,2
83,206,239,178,210,142,184,102,129,102,107,101,
78,93">
<AREA SHAPE="poly" HREF="/Phantom/
index.html"
COORDS="209,107,204,129,240,174,282,201,386,
240,386,366,404,368,477,360,476,255,460,248,460
,241,464,240,463,210,476,208,475,180,469,149,45
5,121,438,100,431,92,394,102,364,103,295,101,29
2,109,282,109,278,119,226,119,222,107,209,108">
<AREA SHAPE="poly" HREF="/Cats/index.html"
COORDS="106,309,114,323,386,321,386,240,286,
203,210,244,179,241,159,246,156,263,162,275,155
,281,139,287,106,309">
<AREA SHAPE="rect" NOHREF
COORDS="0,0,454,511">
</MAP>
```

**Ejemplo de imágen mapa.**

**TABLA 2**

| | |
|---|---|
| <BIG>...</BIG> | Fuente grande respecto al resto |
| <SMALL>...</SMALL> | Fuente pequeña respecto al resto |
| <SUB>...</SUB> | Subíndice |
| <SUP>...</SUP> | Superíndice |
| <P ALIGN=LEFT\|RIGHT\|CENTER> | Alineación de párrafos |
| <EMBED SRC= HEIGHT= WIDTH= ...> | Incrustación de objetos |
| <FONT COLOR=#rrggbb>...</FONT> | Color de la fuente de letra |

**Nuevas extensiones de Netscape.**

jerarquía de zonas calientes que permite la elaboración de figuras complejas. Es decir, si dos zonas se superponen, se tomará como válida en la zona de superposición la que antes figure en el mapa.

Aun con estas facilidades, se ha incorporado un atributo más a la propuesta inicial de Spyglass, el atributo *NOHREF*, que indica que la zona caliente asociada debe ser tambien insensible. De esta forma es posible realizar "agujeros" en la imagen como por ejemplo, la definición de la siguiente "rosquilla":

*<MAP NAME=rosquilla>*
*<AREA SHAPE=CIRCLE COORDS=75,75,50 HREF=rosquilla.html>*
*<AREA SHAPE=CIRCLE COORDS=75,75,25 NOHREF>*
*</MAP>*

En el listado 2, se puede ver un ejemplo muy instructivo que emplea una imagen con mapas tanto en el cliente como en el servidor y cuyo mapa en el cliente hace uso de diferentes *shapes* y del atributo *NOHREF*. La imagen puede observarse en la figura 1.

## NUEVAS EXTENSIONES DE NETSCAPE

Ya en el tercer artículo dedicado a HTML se trataba una serie de nuevas etiquetas propias de Netscape, propuestas como extensiones a la, en aquel momento vigente, versión 2.0 de

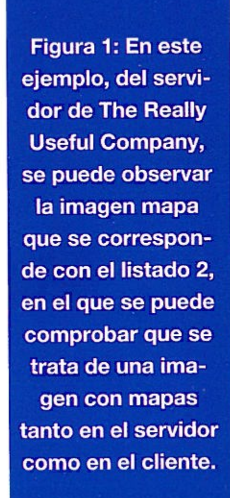
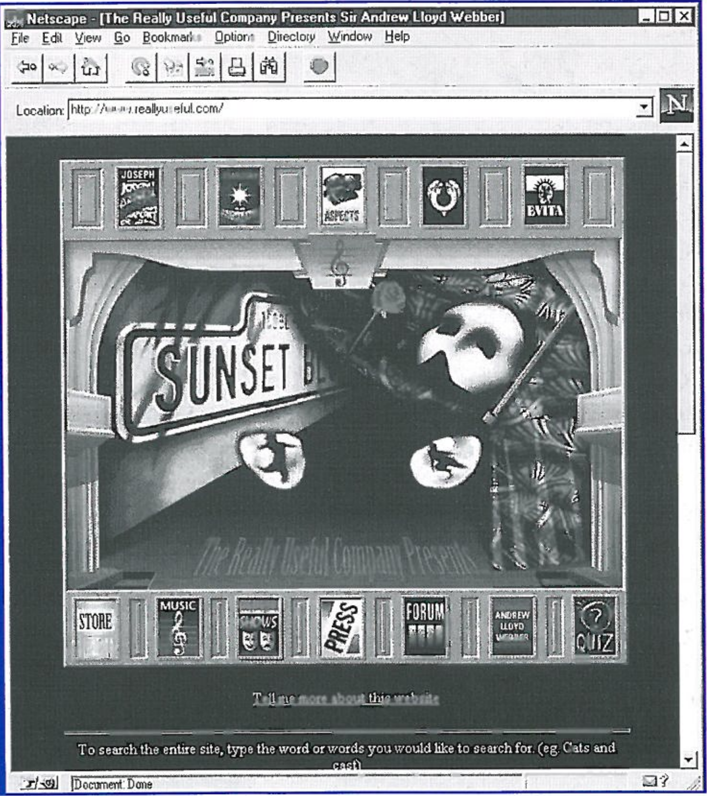


**Figura 1: En este ejemplo, del servidor de The Really Useful Company, se puede observar la imagen mapa que se corresponde con el listado 2, en el que se puede comprobar que se trata de una imagen con mapas tanto en el servidor como en el cliente.**



HTML. Netscape, continuando en la línea de ampliación de estilos físicos, propuso también un conjunto de nuevas etiquetas paralelamente a la discusión de la versión 3.0 de HTML (sin entrar en la discusión referente al apartado de tablas, que se trató en aquel artículo). Así pues, en esta nueva tanda, se incluyen etiquetas destinadas a alterar la fuente de escritura, el funcionamiento del separador de párrafos, color de fuentes e incrustación de objetos:

- *<BIG>...</BIG>*: Presenta el texto, que rodea con una fuenta de mayor tamaño que la empleada hasta el momento. La única ventaja que aporta es que, si se desea cambiar globalmente el tamaño de las fuente de letra de un documento, no es necesario cambiar el atributo *SIZE* de la etiqueta *<FONT>* que se empleaba hasta el momento, en todos los casos en que aparece en el documento, sino que ahora basta con hacerlo en los lugares por defecto.
- *<SMALL>...</SMALL>*: Actúa de forma análoga a la etiqueta *<BIG>*, en este caso, reduciendo el tamaño de la fuente.
- *<SUB>...</SUB>*: Esta nueva etiqueta sí tiene una utilidad considerable al forzar a todo lo que engloba a que aparezca como subíndice. Para ello, si es posible, emplea, además, un tipo de letra más reducido.

**LISTADO 3**

```
<html>
<head><title>VDOPhone</title></head>
<body bgcolor="#ffffff" text="#003000">
<center>
<img src="/gfx/vdophone.gif" alt="[VDOPhone]">
<br>
<table border=7><tr><td align="center">
<i>50 years in the making...</i><br>
<embed src="/vdofiles/phone.vdo" width=160
height=128 stretch="true" autostart="false">
<tr><td>
Clip courtesy of <a href="http://www.cummings-
video.com/home/" target="_blank">Cummings
Multimedia Entertainment</a>
</td></tr></table>
<br>
</center>
(...)
</body>
</html>
```

**Ejemplo de uso de la etiqueta EMBED**

**TABLA 3**

| | |
|---|---|
| <TEXTAREA WRAP=OFF\|VIRTUAL\|PHYSICAL> | Atributos de justificación del texto |
| <FORM ENCTYPE="multipart/form-data"> | Tipo MIME para envío de ficheros |
| <INPUT TYPE="FILE"> | Tipo INPUT para envío de ficheros |

**Extensiones para forms.**

- *<SUP>...</SUP>*: Es la etiqueta análoga a *<SUB>* que se debe emplear para la escritura de super-índices.
- *<P>*: Se propone renovar esta etiqueta, tan antigua como la que más, mediante la adición de los valores *left*, *right*, y *center* del su atributo extendido *ALIGN*.
- *<FONT>...</FONT>*: A la nueva etiqueta *<FONT>* propuesta en el conjunto anterior y de uso ya tan difundido, se le incorpora la posibilidad de elección del color del texto a presentar. Si bien esta posibilidad ya se había incorporado a la etiqueta *<BODY>*, los colores definidos mediante esta última afectaban a todo el conjunto del documento, dado que esta etiqueta no puede aparecer más que una vez en cada una de sus formas de comienzo y fin. Los colores, nuevamente, se indican en rgb [Echeva-5].
- *<EMBED>*: Otro de los aspectos de mayor controversia en el mundo del WWW es la posibilidad de incrustación de objetos en los documentos HTML. La primera propuesta de Netscape al respecto es la incorporación de esta etiqueta al lenguaje, de la que podrán hacer uso los *plug-ins* con los que se amplíe la funcionalidad del browser. Por ello, el conjunto de atributos de la etiqueta *<EMBED>* es arbitrario y dependiente de la aplicación. Como se recordará de [Echeva-3], esta es la misma filosofía que se empleaba para la inclusión de *applets* Java en las páginas HTML mediante la atiqueta *<APPLET>* aunque, en aquella ocasión, la transmisión de parámetros se realizaba mediante una etiqueta adicional *<PARAM>* cuyo atributo *NAME* identificaba los parámetros. En este caso, la etiqueta *<EMBED>* aceptará los siguientes atributos:

***SRC***: EL único atributo de uso obligatorio, especifica el URL del objeto incrustado.

***HEIGTH***: Ancho al cual se ajustará la dimensión vertical del objeto incrustado (si procede).

***WIDTH***: Ancho al cual se ajustará la dimensión horizontal del objeto incrustado (si procede).

En el listado 3 se puede apreciar el código HTML asociado a la figura 6, en la que aparece un vídeo incrustado en un documento gracias al *plug-in VDOLive*.

## EXTENSIONES PARA LA PRESENTACIÓN DE FORMS

Entre las nuevas propuestas de Netscape, también se encuentra una muy esperada dirigida a una mejor presentación de las áreas de texto de los

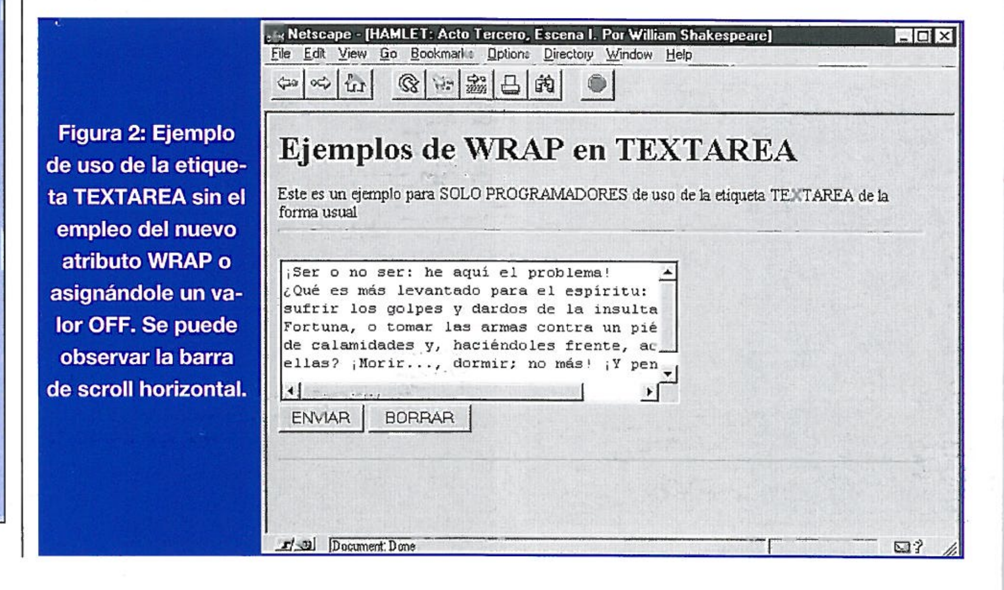



**Figura 2: Ejemplo de uso de la etiqueta TEXTAREA sin el empleo del nuevo atributo WRAP o asignándole un valor OFF. Se puede observar la barra de scroll horizontal.**



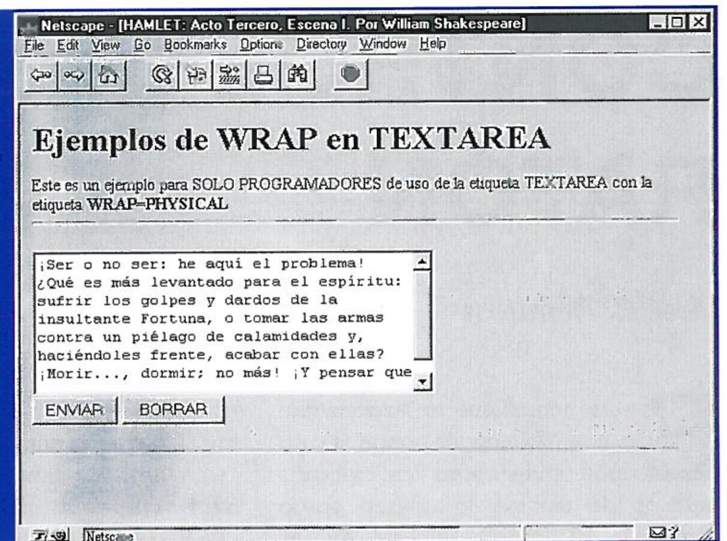



Figura 3: En esta imagen se puede ver el mismo texto de la figura 1, tecleado en un form a cuya etiqueta textarea se le ha añadido la etiqueta WRAP=PHYSICAL. Se puede apreciar cómo la barra de scroll horizontal ha desaparecido.

formularios en pantalla. Se trata de la incorporación del *Wrapping* a los textos introducidos por el usuario. Esta función consiste simplemente en la posibilidad de dividir automáticamente el texto introducido por el usuario en diferentes líneas que se ajusten al ancho de la ventana de texto. Recuérdese que, hasta el momento, mientras no se pulsara la tecla ENTER, todo lo que escribía el usuario en un campo *TEXTÁREA* [Echeva-1], aparecía en una sola línea y, si el tamaño de esta excedía el de la ventana de texto, se podía realizar un desplazamiento del mismo a través de una barra de scroll horizontal (figura 2).

Ahora, se añade un nuevo atributo *WRAP* a la etiqueta *<TEXTAREA>* que puede tomar los valores:

- *OFF*: El campo de texto se comportará de la manera habitual descrita, es decir, no habra *wrapping* y las líneas aparecerán exactamente como se teclean, por lo que equivale a no poner nada (figura 2).
- *VIRTUAL*: El browser realizará una partición en pantalla de las líneas que teclee el usuario con fines estéticos, de la misma forma que ocurre al teclear en un procesador de textos. Sin embargo, el texto enviado será fiel a lo tecleado, es decir, no se habrán insertado caracteres de avance de línea (figura 3).
- *PHYSICAL*: El comportamiento de la pantalla será el mismo que en el caso anterior. No obstante, en esta ocasión la información enviada sí contendrá caracteres añadidos de retorno de línea (figura 3).

## ENVÍO DE FICHEROS MEDIANTE FORMS

Otra de las cuestiones más demandadas por el mundo WWW es la necesidad del envío de ficheros en el sentido cliente-servidor, lo que se ha dado en denominar *HTTP FILE UPLOAD* . La propuesta, que procede de Xerox PARC, se implementa mediante una nueva extensión del mecanismo de forms. Para ello, se añade un nuevo tipo de campo a la etiqueta *<INPUT>*: el tipo *file*. De igual manera, se añade un nuevo tipo MIME para la información enviada: el tipo *multipart/form-data* que se especifica mediante otro nuevo atributo *ENCTYPE* que, en este caso, se incluye en la etiqueta *<FORM>*.

Así, un form destinado al envío de ficheros al servidor se definiría de una forma análoga a:

```
<FORM ENCTYPE="multipart/form-
data"
ACTION="/cgi-bin/getfile"
METHOD=POST>
Enviar el fichero:
<INPUT NAME="filesent" TYPE="file">
<INPUT TYPE="submit"
VALUE="Enviar">
<INPUT TYPE="reset" VALUE="Borrar">
</FORM>
```

La implementación de Netscape de estas nuevas características es muy vistosa, ya que junto al campo de texto destinado al nombre del fichero, aparece un botón para examinar el propio sistema de ficheros (figura 4), lo cual se realiza a través de una ventana proporcionada por el entorno gráfico anfitrión (figura 5). Por supuesto, como en todo form, se dispone de una parte HTML en el cliente y otra CGI en el servidor encargada de procesar la información. Esta última parte se verá en los correspondientes artículos dedicados a CGI.

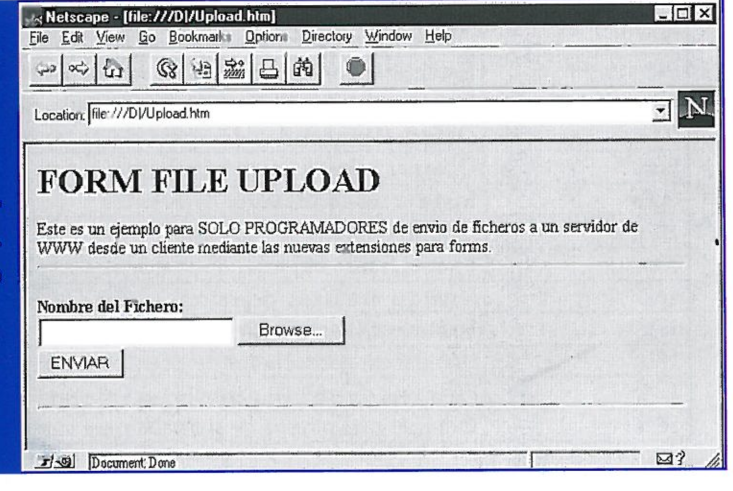



Figura 4: Aspecto que presenta el nuevo valor FILE del atributo INPUT para el envío de ficheros a través de un formulario.


## BIBLIOGRAFÍA

Las referencias incluidas en el texto son a artículos del autor publicados en la revista según:

[Echeva-1], "El lenguaje HTML (II)", núm. 17.
[Echeva-2], "El lenguaje HTML (I)", núm 16.
[Echeva-3], "Los Applets en Java", núm 23.
[Echeva-4], "La interfaz CGI: descripción avanzada", núm 25.
[Echeva-5], "HTML (III). Documentos profesionales", núm 18.


## CONTACTAR CON EL AUTOR

Para cualquier duda, comentario, sugerencia o crítica, se anima al lector a que se ponga en contacto con el autor mediante la sección Correo del Lector o, preferiblemente a través de:

E-mail Internet: echeva@dit.upm.es
E-mail CompuServe: 100646,2456
http://highland.dit.upm.es:8000

# CONTROL DEL RATÓN

*José C. Remiro*




En la actualidad es impensable que un PC no disponga de un ratón, pues todos los entornos gráficos de usuario lo utilizan como periférico de entrada. Incluso en las aplicaciones orientadas a texto el usuario espera poder utilizar este dispositivo. En este artículo se explicará cómo se puede incluir el ratón como dispositivo de entrada en un programa.

### LA INTERFAZ DEL RATÓN

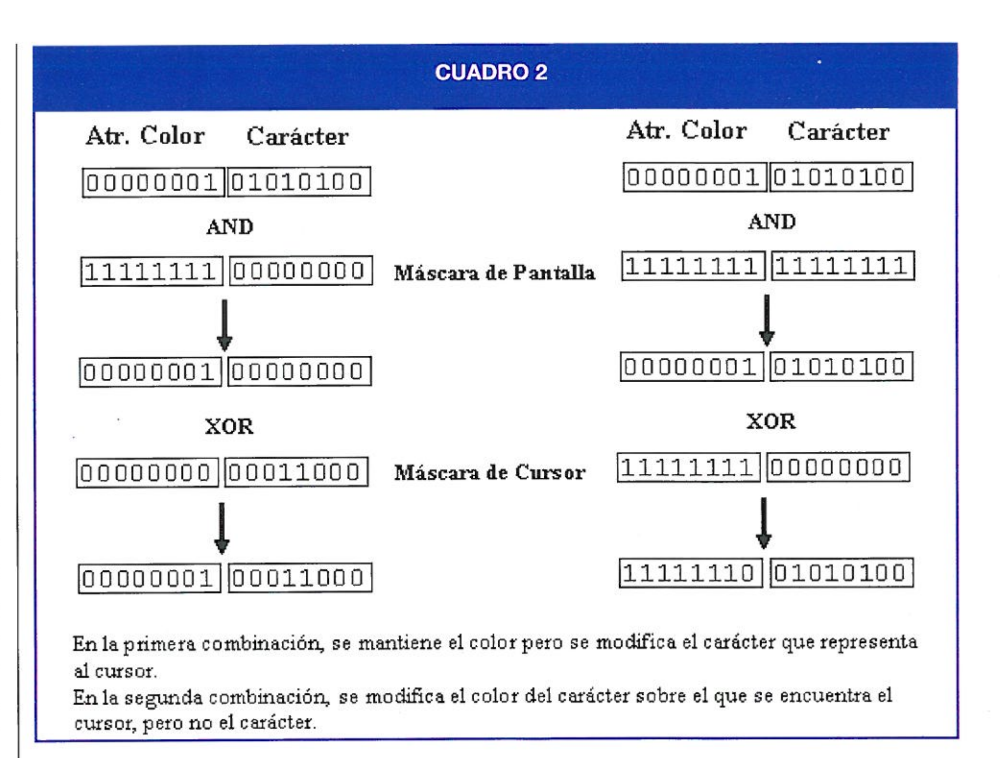
Desde hace tiempo Microsoft definió un API que proporcionaba una serie de funciones para que un programa pudiera utilizar el ratón. Como es natural, otros fabricantes han aceptado esta interfaz para asegurar la compatibilidad. Normalmente, la interfaz de funciones se instala al arrancar el sistema, mediante un controlador de dispositivo (mouse.sys) o bien mediante un programa residente en memoria (mouse.com).

La forma de llamar a una función del BIOS o del DOS no difiere en nada de una llamada a la interfaz del ratón, pues ésta se realiza a través de una interrupción especial: la interrupción 51 (33h). Cada vez que se desee utilizar una función asociada a la interfaz del ratón basta con indicar su número en el registro AX. El resto de registros pueden ser necesarios para alguna función con el fin de pasarle alguna otra información. Una vez ejecutada la llamada a la función en los registros se obtendrá la correspondiente información. En el cuadro 1 se muestran las funciones más importantes del controlador del ratón (solamente se indican las más importantes, pero existen hasta 53 funciones en la versión 8).

**CUADRO 1**

| Función | Descripción |
|---|---|
| 00h | Inicializar el controlador del ratón |
| 01h | Visualizar el cursor del ratón en pantalla |
| 02h | Ocultar el cursro del ratón |
| 03h | Obtener el estado del ratón |
| 04h | Mover el cursor del ratón |
| 05h | Obtener las veces que se ha pulsado un botón |
| 06h | Obtener las veces que se liberó un botón |
| 07h | Establecer zona horizontal de movimiento del cursor |
| 08h | Establecer zona vertical de movimiento del cursor |
| 0Ah | Definir el cursor de ratón para modo texto |
| 0Bh | Obtener valores del movimiento |
| 0Ch | Instalar un controlador de eventos |
| 10h | Establecer zona de exlusión |
| 13h | Fijar el límite superior de aumento de velocidad |
| 16h | Guardar el estado del ratón |
| 17h | Restaurar el estado del ratón |
| 18h | Instalar un controlador de eventos alternativo |
| 1Dh | Fijar una página de pantalla para el cursor del ratón |
| 1Eh | Obtener una página de pantalla para el cursor del ratón |

**Actualmente el ratón se ha hecho casi indispensable para poder utilizar un programa. En este artículo se describe la forma de poder controlarlo. Una vez estudiadas las principales técnicas de programación, seguimos con el desarrollo de ciertas aplicaciones.**




En la primera combinación, se mantiene el color pero se modifica el carácter que representa al cursor.
En la segunda combinación, se modifica el color del carácter sobre el que se encuentra el cursor, pero no el carácter.

La información que se espera obtener de un ratón consiste en saber si se ha pulsado uno de sus botones (puede disponer de dos o tres) y la posición en la que se encuentra el cursor del ratón, lo que permite al programador saber en cada momento sobre qué objeto de la pantalla se encuentra situado, y de esta forma seleccionar la secuencia de acciones correspondientes.

El ratón aparece representado en pantalla mediante un cursor, pero las coordenadas donde se encuentra situado dicho cursor se interpretan siempre en relación a una pantalla gráfica virtual. De esta forma, si se desea trasladar dichas coordenadas en una aplicación que utilice una pantalla de texto, se debe hacer corresponder las coordenadas gráficas con las coordenadas utilizadas por una pantalla de texto (línea, columna). Puesto que cada línea y columna se corresponden con ocho puntos gráficos, se deberá dividir por ocho cada una de las componentes que forman las coordenadas gráficas. Esta operación puede realizarse de forma más eficiente si se desplazan hacia la derecha 3 bits de cada una de las componentes.

## INICIALIZAR EL RATÓN

La primera tarea que debe realizar un programa que desee utilizar el ratón como dispositivo de entrada es comprobar si hay conectado uno al sistema, y en caso de existir inicializarlo, guardando en este caso los valores que tenia con anterioridad para, al finalizar el programa, restaurar dichos valores.

La primera de las acciones puede realizarse a través de la función 00h (inicialización del ratón). Tras la llamada a esta función puede descubrirse si hay instalado un controlador para el ratón (puede ocurrir que no esté conectado físicamente el ratón). El registro AX contendrá el valor FFFFh si el controlador se encuentra instalado. En otro caso, dicho registro contendrá el valor 0000h.

Mediante la función 16h puede guardarse el estado del ratón, con lo que se puede restaurar el estado anterior al de inicialización del ratón.

Al iniciar el ratón se procede a situar el cursor de éste en el centro de la pantalla. Se define dicho cursor como un rectángulo, seleccionándose la página 0 de pantalla para la salida del ratón y por último desaparece el cursor del ratón.

## CARACTERÍSTICAS DEL CURSOR

Una vez que se ha producido la inicialización del ratón es posible utilizar las funciones 01h y 02h, que muestran y ocultan, respectivamente, el cursor del ratón.

**Una vez que se ha producido la inicialización del ratón es posible utilizar las funciones 01h y 02h**

Como se comentó con anterioridad, al realizar la inicialización del ratón el cursor, antes de desaparecer, aparecía en el centro de la pantalla. Pero puesto que el controlador sigue los movimientos del ratón aún cuando el cursor de éste no sea visible, cuando se proceda a su visualización es posible que no aparezca en el mismo lugar donde se mostró inicialmente. Esto sigue siendo válido cuando se utiliza la función 02h para ocultar el cursor.

El uso de las anteriores funciones no es intensivo si se utilizan las funciones DOS y BIOS para la salida en pantalla, pues será suficiente con llamar a la función 01h tras llamar a la función 00h para mostrar el cursor y, posteriormente, llamar a la función 02h para ocultar el cursor antes de salir del programa. Sin embargo, si se desea evitar la lentitud de las rutinas DOS y BIOS para la salida por pantalla y se opta por efectuar la salida en pantalla directamente en la memoria RAM de vídeo habrá que utilizarlas de forma reiterada.

Cuando se mueve el cursor del ratón y éste es visible, se guardan los atributos y el carácter sobre el que se encuentra el cursor, de tal forma que cuando se mueva de nuevo el cursor, el carácter sobre el que se encontraba anteriormente pueda ser restaurado. Estas acciones se realizan gracias a que durante la instalación del controlador del ratón se redirecciona a una rutina propia del controlador el vector de interrupción 10h (BIOS de vídeo). Sin embargo, si se utiliza la salida en pantalla directamente sobre la RAM de vídeo, no se guarda la anterior información, por lo que el programa antes de realizar una salida deberá ocultar el cursor, posteriormente realizar la salida en pantalla y por último mostrar el cursor. Por último, en relación a estas dos funciones, hay que señalar que las llamadas deben estar equilibra-

das de tal forma que si se llama tres veces consecutivas a la función 02h, para ocultar el cursor habrá que llamar igual número de veces a la función 01h.

En cuanto a la apariencia del cursor en pantalla, cuando ésta se encuentra en modo texto se pueden seleccionar varias. En primer lugar, se puede elegir entre un cursor hardware, en este caso el cursor aparecerá intermitentemente y solamente se podrá seleccionar la línea inicial y final del carácter que esté ocultando, o bien, un cursor software, que puede representarse de varias formas dependiendo de la máscara de pantalla y de la máscara de cursor. A continuación se describirá como funcionan.

Como se sabe, un carácter junto con sus atributos de visualización se representan con una secuencia de 16 bits, los ocho de menor peso para representar el código ASCII del carácter a representar y los otros ocho restantes para especificar los colores de primer plano y de fondo (se supone que se dispone de una tarjeta color). Las máscaras de pantalla y de cursor son dos valores de 16 bits. Cada vez que se modifica la posición del cursor, el controlador del ratón aplica a la representación del carácter sobre el que se sitúa una operación *and* con la máscara de pantalla y a éste resultado se le aplica una operación *xor* (OR exclusivo) con la máscara de cursor. El resultado obtenido será la representación final del carácter apuntado por el cursor. Aunque el conjunto de posibilidades es bastante amplio, normalmente se suele utilizar o bien, un cursor fijo independientemente del carácter sobre el que se encuentre, o bien el carácter sobre el que está situado el cursor modifica su color dependiendo del color inicial del carácter o el cursor cambia de color dependiendo del color del carácter sobre el que se sitúa, pero se muestra siempre mediante el mismo carácter. En el cuadro 2 se muestran las anteriores combinaciones junto con las máscaras que se utilizan. La función que controla el aspecto del cursor del ratón es la 0Ah.

Para indicar que se desea utilizar un cursor hardware debe ponerse el registro BX a 1, indicando en el registro CX la línea inicial del cursor y en el registro DX la línea final del cursor. Los valores permitidos en estos registros cuando se utiliza una tarjeta VGA son de 0 a 7, siendo interpretados directamente por la BIOS. Si el registro BX se pone a 0, entonces se indicará que se desea utilizar un cursor software. En este caso, dentro del registro DX se pondrá la máscara de pantalla y dentro del registro CX se incluirá la máscara de cursor.

**Además, se puede seleccionar una zona de la pantalla donde se podrá utilizar el ratón**

### DESPLAZAMIENTOS Y ZONAS

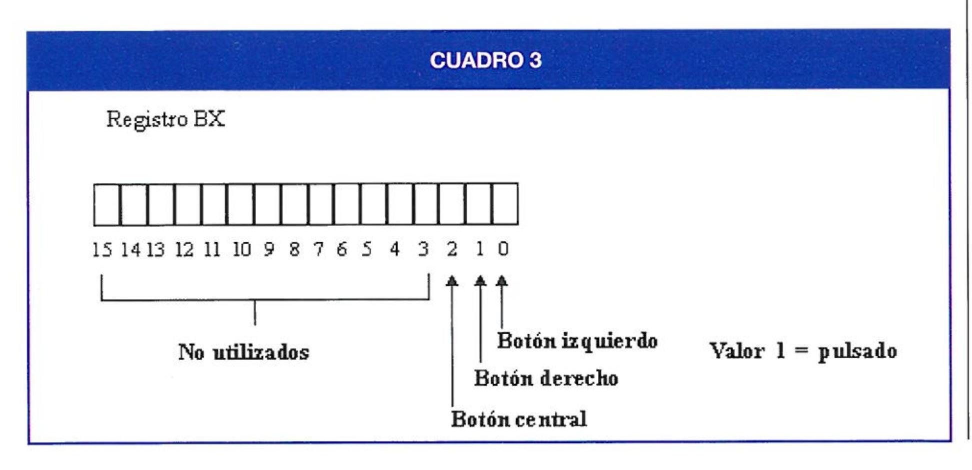
Otra de las posibilidades ofrecidas por el controlador del ratón es poder desplazar el cursor a una posición determinada de la pantalla. Utilizando la función 04h, indicando en el registro CX la columna y en el registro DX la fila, se desplazará el cursor a dichas coordenadas. Estas deben referirse a la pantalla virtual del ratón, es decir, si se está trabajando con una pantalla de texto las coordenadas de ésta deben ser multiplicadas por ocho.

Además, se puede seleccionar una zona de la pantalla donde se podrá utilizar el ratón, quedando el resto fuera de la acción del ratón. Cuando se procede a la inicialización del ratón a través de la función 00h toda la pantalla es accesible para éste, pero con las funciones 07h y 08h se pueden limitar las zonas de movimiento horizontal y vertical respectivamente. Bastará con utilizar la función correspondiente junto con los valores mínimo y máximo (tanto para la zona horizontal como vertical) en el registro CX y en el registro DX. Una vez definida esta zona, el cursor del ratón se situará dentro de la zona definida y no podrá salir de estos límites. De nuevo las coordenadas estarán referidas a la pantalla virtual asociada al ratón.

A través de la función 10h se puede definir, en cualquier momento, una zona rectangular donde no es visible el cursor. Solamente se puede especificar una zona de forma simultánea. Junto con el número de función en el registro AX, se debe especificar la esquina superior izquierda en los registros CX y DX, en los registros SI y DI se especificará la esquina inferior derecha, la fila se indicará en DX y DI, mientras que la columna se especificará en CX y SI. Definir una zona donde el cursor no es visible es útil cuando hay que reconstruir una figura de tipo rectangular y se esté utilizando el acceso directo a la RAM de vídeo. Se puede definir la zona de no visualización como la zona en la que reconstruirá la figura. De esta forma el ratón no desaparecerá en el resto de la pantalla. Tras reconstruir la figura se realizará una llamada a la función 00h o 01h para que el cursor vuelva a aparecer en la totalidad de la pantalla.

Por último, se hablará de la velocidad del ratón. La distancia recorrida por el ratón se mide en *Mickeys* que se corresponde en un ratón con una resolución de 400 puntos por pulgada a 1/400 pulgadas (0,0635 mm). Se entiende por velocidad de movimiento del ratón la relación entre el movimiento del cursor y los

**CUADRO 3**



puntos desplazados en la pantalla virtual del mismo. Con la función 0Fh se puede ajustar esta relación, de tal forma que se indicará el número de *Mickeys* que se desplazarán horizontal y verticalmente en el registro CX y DX respectivamente. De todas formas, cuando se realiza la llamada a la función 00h, se establecen para una pantalla de tipo texto 8 *Mickeys* para el movimiento horizontal y 16 para el movimiento vertical.

## EL CONTROL DEL RATÓN

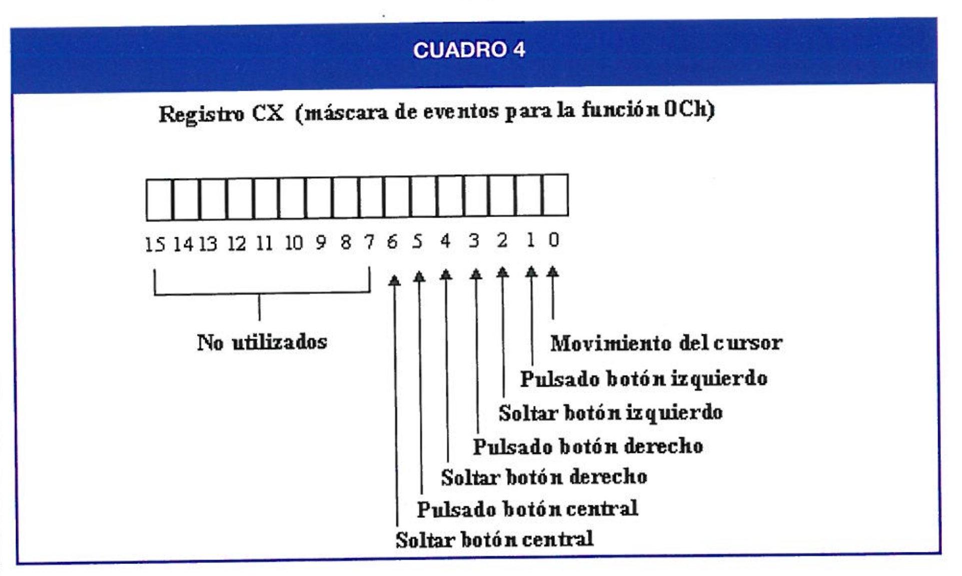
Hasta ahora se han descrito todas las características que se pueden modificar en el ratón, pero se ha dejado para finalizar el control del ratón. Por éste se entenderá conocer en un momento dado dónde se encuentra situado su cursor y si el estado de los botones ha sido modificado para así tomar las acciones oportunas.

Hay dos formas de controlar el ratón, utilizar la función 03h o bien instalar uno o varios controladores de eventos utilizando la función 0Ch.

La función 03h devuelve en el registro BX la información sobre el estado de los botones, mientras que en los registros CX y DX se almacena la coordenada horizontal y vertical respectivamente. El bit 0 del registro BX indicará si se ha pulsado o no el botón izquierdo (el bit con valor 1 equivale a pulsado), los bits 1 y 2 devuelven la misma información para los botones derecho y central (si existe) respectivamente (ver cuadro 3).

La principal ventaja de utilizar la función 03h es la facilidad para programar el ratón, siendo el principal inconveniente el tener que utilizar un bucle para comprobar que se produce el evento esperado (cambio de zona, estado de un botón etc...). Por tanto, la CPU mientras dure este bucle estará ocupada consultando constantemente el estado del ratón.

CUADRO 4




## El controlador de eventos no puede terminar con la instrucción IRET

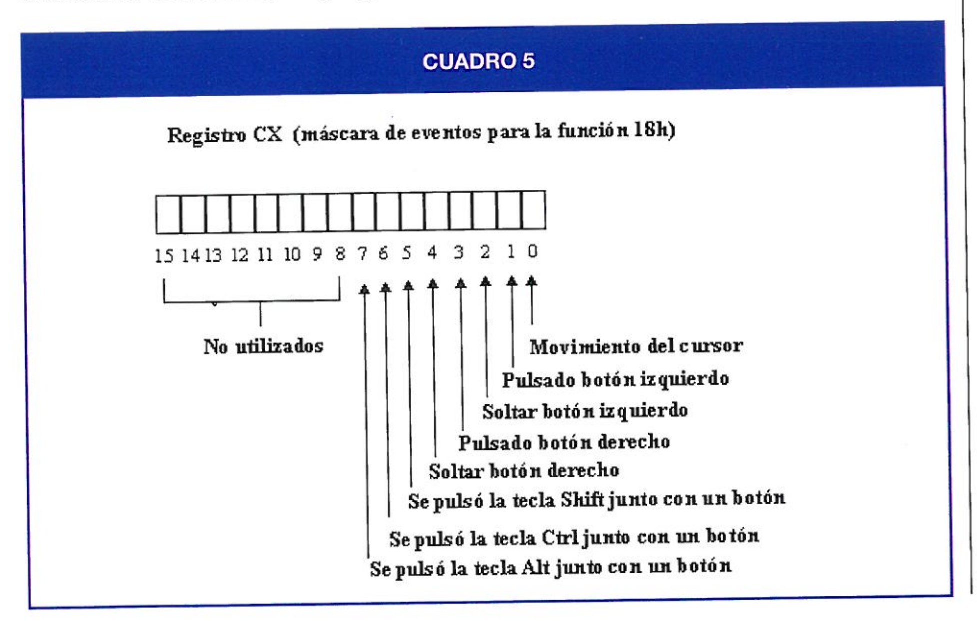
Otra forma de controlar el ratón consiste en instalar un controlador de eventos utilizando la función 0Ch. El controlador de eventos no es más que una rutina que se activa gracias a que el controlador del ratón le comunica a través de una interrupción que se ha producido un determinado suceso, representado en la máscara de eventos. El controlador de eventos será el encargado de tratar la información que provocó la interrupción por parte del controlador del ratón. De esta forma es posible controlar el ratón sin necesidad de ocupar constantemente la CPU con esta tarea.

La información que comunica el controlador del ratón al controlador de eventos se encuentra en los registros del procesador. Así, en el registro AX se encontrará la bandera de eventos, que indica el suceso que ha provocado la llamada, en el registro CX se encuentra la coordenada horizontal, mientras que en DX se almacena la coordenada vertical. En cuanto a la programación de un controlador de eventos a través de la función 0Ch hay que tener en cuenta una serie de requisitos.

Aunque con anterioridad se dijo que el controlador del ratón llamaba al controlador de eventos a través de una interrupción, ésta no se realiza mediante la instrucción INT, sino por medio de una llamada lejana. Por lo tanto, el controlador de eventos no puede terminar con la instrucción IRET, sino con una instrucción RET. Además, habrá que tener en cuenta que el registro DS deberá apuntar al segmento de datos del controlador del ratón y no al segmento de datos del programa interrumpido.

Es posible, con ayuda de la función 18h, instalar un controlador de eventos que responda a eventos del teclado

CUADRO 5



EL RATÓN SE HA CONVERTIDO EN UN ELEMENTO IMPRESCINDIBLE DEL PC



junto con eventos producidos a través del ratón, aunque solamente se pueden controlar las teclas Shift, Control y Alt. En el cuadro 5 se encuentra la máscara de eventos para esta función.

## EJEMPLOS

Los programas que acompañan a este artículo pretenden mostrar cómo incluir el ratón como dispositivo de entrada en una aplicación. En la unidad *raton1.pas* se han incluido una serie de procedimientos para inicializar el ratón, mostrar y ocultar el cursor y obtener su estado.

El programa *prbrat.pas* utiliza la anterior unidad para mostrar en pantalla el número de botones del ratón, la coordenada donde se encuentra situado su puntero y los botones que se pulsan en cada momento. Si se observa este programa se verá que se trata de un típico ejemplo para utilizar el ratón a través de la función 03h (ver procedimiento *obten_estado*). Se produce una llamada de forma continuada hasta que se pulsa una tecla, visualizando en pantalla la información correspondiente al estado del ratón.

No se han incluido en esta unidad otros procedimientos y funciones para el resto de las funciones del ratón debido a que su construcción es muy similar a las incluidas. Basta con introducir el número de la función, junto con los valores de sus parámetros (si los tiene), en los registros del procesador para obtener el efecto deseado. Si se desea modificar el aspecto del ratón mediante las máscaras de pantalla y de cursor se puede modificar el procedimiento *inicia_raton* incluyendo los valores de las máscaras en los registros CX y DX respectivamente.

En la unidad *raton2.pas* se han incluido dos procedimientos que permiten instalar y trabajar con un controlador de eventos junto con la rutina en ensamblador *auxensam.asm*. La solución presentada es particular para el compilador Turbo Pascal y especialmente se debe prestar atención si se desea implementar estas rutinas en otro lenguaje a la implementación que haga éste del control de programa. Así, antes de llamar a un procedimiento o función Turbo Pascal, los parámetros se apilan en el *stack* por su orden de declaración. Antes de que el procedimiento o la función retorne el control, eliminan del stack todos los parámetros. Puesto que solamente se van a utilizar parámetros por valor, bastará con apilar en el *stack* los diferentes valores que se desean transferir al procedimiento.

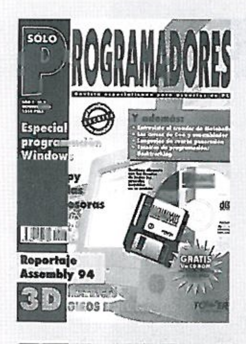
El controlador de eventos está representado por la rutina *auxensam*, que a su vez se ayuda del procedimiento *controla_eventos* perteneciente a la unidad *raton2.pas*. La rutina en ensamblador asegura que los requisitos para un controlador de eventos, que se describieron con anterioridad, se cumplen. Así, y tras haber indicado al controlador del ratón mediante la función 0Ch (procedimiento *instala_controlador*), que el controlador de eventos es *auxensam* (indicando el segmento y el desplazamiento de la rutina), cada vez que se modifica el estado del ratón se llamará a dicha rutina, que a su vez apilará los valores de los registros en el *stack* y procederá a la llamada del procedimiento *Controla_eventos*. Una vez terminado este procedimiento se desapilan del *stack* los valores anteriormente guardados y se finaliza la rutina.

Puesto que el procedimiento en Pascal puede llevar demasiado tiempo en ejecutarse y puede ocurrir que el estado del ratón cambie de nuevo cuando se está tratando una llamada anterior, se ha incluido una variable (*ocupado*) que indica si se está procesando una llamada anterior o no.

El procedimiento controla_eventos no hace más que interpretar la información que ha recibido de la rutina, completando el conjunto de eventos que se encuentran almacenados en la variable global *evento_actual* y la posición actual del cursor del ratón en las variables *pos_x* y *pos_y*. Consultando estas variables globales se puede decidir realizar una determinada acción fuera del procedimiento.

# CÓMO SUSCRIBIRSE A













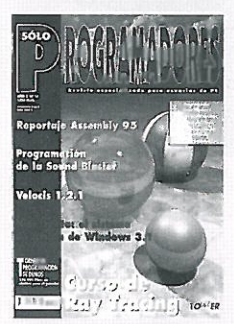








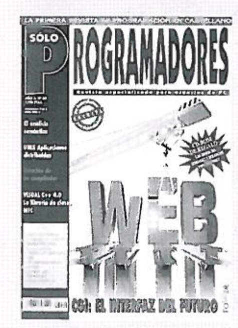

















eo suscribirme a la revista **SÓLO PROGRAMADORES** acogiéndome a la siguiente modalidad:

☐ Suscripción: 1 año (12 números) **por sólo** 11.950 ptas. (ahorro 20%). ☐ Estudiantes carreras técnicas: 8.950 ptas. (ahorro 40%)

**STA OFERTA ANULA LAS ANTERIORES, DESCUENTOS NO ACUMULABLES.**

nbre y apellidos.............................................................................Domicilio..........................................................................................

lación...............................................................C.P ..................Provincia...........................Telf.................................Profesión.....................

**RMA DE PAGO:**

☐ Con cargo a mi tarjeta VISA nº ___ ___ ___ ___

Fecha de caducidad de la tarjeta.................Nombre del titular, si es distinto.........................................................................

☐ Domiciliación bancaria.

Señor Director del banco ...........................................

Población .....................................................................

Ruego a vd. que se sirva cargar en mi ☐ cuenta corriente ☐ libreta

horro número........................................................

ibo que le será presentado por TOWER COMMUNICATIONS, S.R.L.

pago de mi suscripción a la revista SÓLO PROGRAMADORES.

| CODIGO CUENTA CLIENTE | | | |
|---|---|---|---|
| ENTIDAD | OFICINA | DC | Nº CUENTA |
| | | | |

**Firma:**

☐ Contra-reembolso del importe más gastos de envío.

☐ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.

☐ Giro Postal (adjunto fotocopia del resguardo).

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Aragoneses, 7
28100 Pol. Ind. ALCOBENDAS (Madrid)

# CHEQUEO A INTERNET

*Fernando J. Echevarrieta*

unix X

**La costumbre, en muchas ocasiones, hace que no se repare en los mecanismos que permiten servicios tan "corrientes" como la videoconferencia, las transferencias intercontinentales de ficheros, conversaciones de larga distancia, acceso a ordenadores remotos, servicios de consulta como el WWW... Simplemente, se usan. En el presente artículo se estudiará de dónde proviene la magia que hace posible Internet: el protocolo IP.**

El mes pasado se comentaba cómo lo que se entiende por TCP/IP es una arquitectura completa de comunicaciones formada por multitud de protocolos. Existían protocolos con funciones muy diversas, unos simples y otros más complejos, unos necesarios y otros reemplazables. Pero de todos ellos hay uno que jamás puede faltar, ya que sin él la palabra Internet carece de sentido: el protocolo IP.

No procede explicar todos los detalles del protocolo en la revista, lo que resultaría en un texto tan aburrido como poco didáctico, pero para los lectores que deseen conocerlo totalmente, se facilita junto a la revista, el RFC791 "INTERNET PROTOCOL".

En el presente artículo se centrará el punto de mira en el protocolo IP con el objetivo de comprender cómo funciona Internet. Pero este estudio se realizará desde una perspectiva más amplia, empleando el modelo OSI como referencia didáctica. No se limitará a una mera descripción del protocolo, sino que se le situará en un entorno explicando de dónde surgen las necesidades que han hecho que el protocolo sea así, introduciendo conceptos como los de MTU y fragmentación de PDUs y, por supuesto, planteando la problemática que debe resolver un nivel de red. Para ello, se comenzará recordando el planteamiento del problema de la comunicación entre máquinas.

## REPASO DE CONCEPTOS SOBRE PROTOCOLOS

Un sistema abierto se define como una serie de niveles funcionales, cada uno de los cuales resuelve un conjunto de necesidades y problemas relacionados con la comunicación. En los límites entre niveles se define una serie de interfaces y en un interior, una serie de protocolos.

Un protocolo queda determinado por el formato de sus *PDUs* (explicadas más adelante) y el servicio que presta al nivel superior, que se refleja en una API de primitivas de servicio que configuran su interfaz de usuario. En definitiva, se está hablando de módulos software que presentarán una API de funciones.

No hay que confundir la idea de protocolo con la de servicio. Un servicio especifica las funciones que debe cumplir el protocolo y, de esta forma, determina una API, pero no dice nada acerca de cómo debe lograr el protocolo este servicio. El protocolo, por su parte, especifica el formato de la información que se va a intercambiar por la red, así como las reglas necesarias para ofrecer un servicio determinado, pero no debe dejar ver a sus usuarios cómo funciona interiormente, de tal forma que un cambio en la implementación del protocolo no afecte a la API definida por el servicio que presta.

Con el objeto de que se pueda calificar como abierto, es fundamental que el modelo sea independiente de la tecnología y se respeten las interfaces. De esta forma, es posible intercambiar un nivel entero o uno de los protocolos sin que afecte al conjunto, ya que la interfaz de servicio será mantenida. En la práctica, como se estudió en el artículo del mes anterior, esta descomposición en niveles se suele tomar como modelo de referencia para el diseño. Pero a la hora de la implementación es común encontrar módulos software que traspasan las fronteras de los niveles teóricos con el objeto de optimizar prestaciones.

En la práctica totalidad de la bibliografía, se denomina *usuario* a cualquier entidad que hace uso de la API de un protocolo. Así, por ejemplo, un usuario de IP puede ser tanto un programa de aplicación como un protocolo de nivel

superior. Cuando se habla de protocolo de nivel superior se suele tratar de una mala traducción de *Higher Level Protocol*, cuando realmente se debería hablar de un protocolo de más alto nivel o, para dejarlo aún más claro, simplemente del protocolo usuario. Esto da lugar a profundas confusiones, ya que el protocolo usuario de un protocolo de nivel *n* no tiene por qué pertenecer al nivel *n+1* del modelo de referencia.

Así, por ejemplo, es común emplear un protocolo como X.25, teóricamente de nivel 3 (realmente tiene funciones de enlace y no es un buen protocolo de nivel 3), dando servicio a otro protocolo de nivel 3, como IP. De esta forma, es posible generar una red virtual (objetivo del nivel) no solamente a partir de redes físicas, sino también a partir de otras redes virtuales con su propio protocolo de nivel 3.

Aún más claro es el ejemplo del protocolo ICMP que realiza funciones de detección de errores asociadas al nivel 3 y utiliza los servicios de IP, es decir, es usuario de IP, encontrándose ambos en el nivel 3. En realidad se considera ICMP como parte de IP, y las especificaciones exigen que toda implementación de IP se vea acompañada de otra de ICMP, pero se ahondará en este ejemplo cuando se estudie el ICMP.

En cualquier caso, y por presentar una homogeneización con la bibliografía existente en castellano, se continuarán empleando los términos "protocolo de nivel superior" y "nivel superior" esperando que, una vez aclarado esto, no dé lugar a confusión en el lector.

Un usuario (en el sentido anteriormente definido) entrega datos al software del protocolo de nivel inferior en forma de unidades denominadas *SDUs* o *Service Data Unit*. La interfaz de cada protocolo puede aceptar un tamaño máximo de *SDU*, por lo que el usuario deberá ser capaz de fragmentar la información de que dispone en varias *SDUs* si es necesario. En el clásico modelo de niveles en torre, las *SDUs* siguen un recorrido vertical pasando de (protocolo o aplicación) usuario a protocolo de nivel inferior.

Las *PDUs*, *Protocol Data Unit*, son la unidades de datos del protocolo e intervienen en una comunicación "horizontal" entre dos entidades del mismo nivel. El software del protocolo construye sus *PDUs* añadiendo sus cabeceras de protocolo a la información que recibe del nivel superior. Así se puede expresar que

*PDU(n)=PDU(n+1)+Cabeceras(n)*

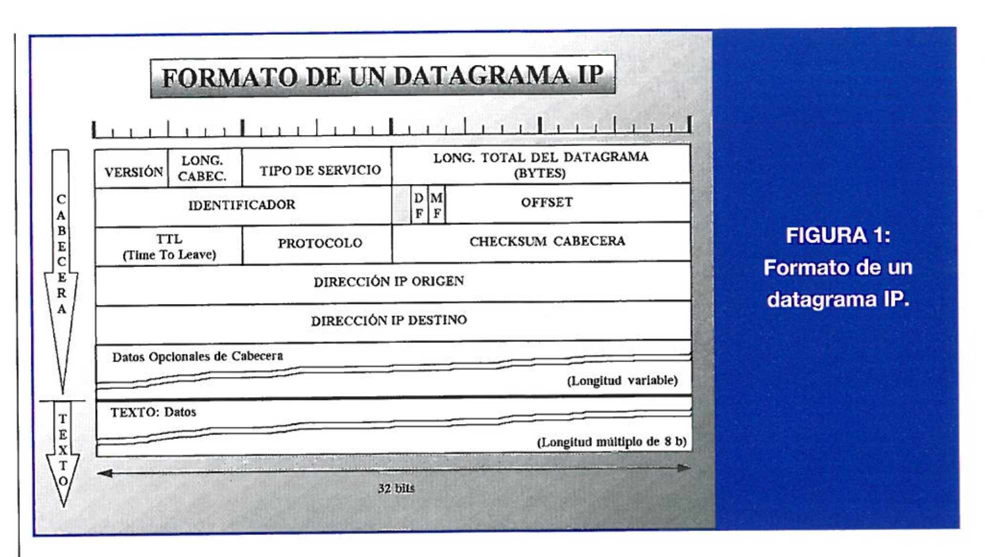



**FIGURA 1: Formato de un datagrama IP.**

Al estudiar un determinado nivel, se abstrae el camino real de la información y se trabaja con la idea de que estas *PDUs* son transmitidas a la entidad homóloga remota directamente, de forma "horizontal". En el proceso real, estas *PDUs* son transmitidas de nuevo al protocolo de nivel inferior en forma de *SDUs* que se transmiten empleando la interfaz definida. Cada uno de los puntos de esta interfaz que se emplean para permitir el paso de una *SDU* de un nivel a otro recibe el nombre de *SAP*, *Service Access Point* (Punto de Acceso al Servicio).

Ésta es la ventaja de los modelos de referencia en niveles, ya que permite obviar el camino real de la información y los problemas derivados de él, que se consideran resueltos.

## FUNCIONES DEL NIVEL INTER-RED

Como se adelantaba en capítulos anteriores, la función del nivel inter-red (terminología TCP/IP), nivel de red o nivel 3 (terminología OSI), es generar una red virtual que interconecte sistemas finales que se encuentren en diferentes redes locales. Es esta red virtual la que recibe el nombre de red inter-red, red internet (con minúscula).

En este nivel de la arquitectura de protocolos se consideran resueltos todos los problemas de los que se ocupan el nivel 1, físico, es decir la transmisión y recepción de señales por el medio físico; y el nivel 2, de enlace, es decir, la de proporcionar una comunicación libre de los errores que se puedan deber al medio físico entre dos máquinas directamente conectadas (lo que recibe el nombre de "enlace").

Así pues, se trata de resolver una comunicación entre sistemas finales, se encuentren o no directamente conectados. Por tanto, una de las funciones fundamentales que debe asumir el nivel de red es la de encaminamiento de la información desde el origen hasta el destino, para lo cual se debe establecer un conjunto de direcciones lógicas o de red. Estas direcciones deben ser independientes de la tecnología de las máquinas involucradas en la comunicación con el fin de que se trate de un modelo de sistemas abiertos (en [Echeva-1] se expuso el modelo de direcciones IP de internet). La comunicación entre máquinas se realiza salto a salto hasta alcanzar el destino. Por ello, cada nodo deberá conocer la dirección de red del destino, así como la del siguiente sistema intermedio.

En definitiva, se podrían señalar como funciones primordiales de un nivel de red las siguientes:

- 1. Generación de la red virtual
- 2. Plan de direccionamiento independiente de la tecnología
- 3. Encaminamiento de la información entre extremos
- 4. Detección de errores

En el artículo del mes pasado se expuso el plan de direccionamiento de

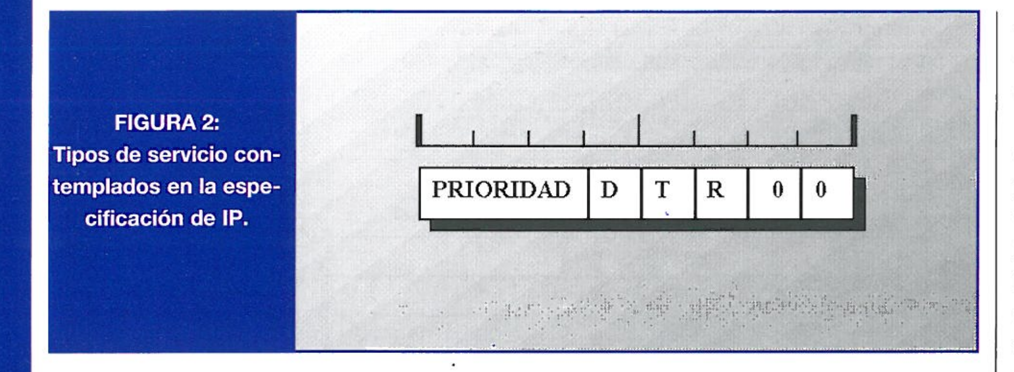


FIGURA 2: Tipos de servicio contemplados en la especificación de IP.

Internet. En esta entrega se expondrá el mecanismo de comunicación definido por IP en internet para la integración de subredes heterogéneas y se dejará para más adelante el estudio del encaminamiento IP y la detección de errores asociada, que se realiza a través del protocolo ICMP.

## EL PROTOCOLO IP

El protocolo encargado de generar la red virtual en Internet es el *Internet Protocol* o *IP*. Por ello, este protocolo es el que determina el formato de los datos que circulan a través de Internet. Pero un protocolo no se limita a definir el formato de los datos, sino que también aporta un conjunto de reglas sobre cómo procesarlos, con el objeto de poder prestar un servicio definido al nivel superior. Y entre las normas que impone un protocolo de nivel 3, la fundamental es la función de encaminamiento.

Como se indicaba en el artículo del número anterior, se trata de un protocolo no orientado a conexión no fiable, lo que significa que puede presentar pérdidas, duplicados y desorden en la recepción de los datagramas. En cualquier caso, se trata de un protocolo denominado *best effort*, es decir, que en lugar de descartar datagramas caprichosamente, "lo hará lo mejor posible" para encaminarlos a su destino.

Las labores de recuperación de los errores se reservan para protocolos de nivel superior, como TCP, aunque a nivel de red se ha definido un mecanismo para la notificación (que no recuperación) de estas situaciones anómalas: el protocolo ICMP.

En la figura 1 se puede observar el formato de un datagrama IP. La representación se ha realizado en filas alineadas según palabras de 32 bits, pero se entiende que un datagrama IP es una secuencia continua de estos bits con una longitud total en bytes.

Un datagrama IP consta de cabecera, con información necesaria para el funcionamiento del protocolo, y datos, que es la parte correspondiente a los datos transportados. La cabecera está formada por una parte fija de 20 bytes y una parte opcional de longitud variable. Los campos de la cabecera se detallan a continuación:

• ***Número de versión***: Todo datagrama IP lleva impreso el número de versión del protocolo. De esta manera es posible actualizar a nivel mundial el mismo de una manera gradual, permitiendo coexistir en internet diferentes versiones de IP.

Si un nodo intermedio recibe un datagrama con una versión de IP que no es capaz de procesar, simplemente lo descarta. Esto no supone ningún error grave, al tratarse de un protocolo no fiable y, de este modo, se evita la posibilidad de interpretar incorrectamente su contenido.

De esta forma, también es posible experimentar con nuevas versiones en desarrollo en un entorno real. Así, por ejemplo, en el anterior artículo de la serie se hacía referencia a la nueva versión IPv6, cuya incorporación no supondrá la parada de toda la red internet para realizar el cambio de protocolo, sino que se podrá incorporar gradualmente coexistiendo con la actual, IPv4.

• ***Longitud de la cabecera***: Dado que la cabecera dispone de una serie de campos opcionales, es necesario indicar su longitud para poder separarla correctamente del texto, razón por la que existe este campo.

• ***Tipo de Servicio***: La semántica asociada a cada uno de los bits que componen el byte que define el tipo de servicio se puede observar en la figura 2. En ella se observa que los primeros 3 bits se emplean para indicar la prioridad del datagrama. Esta prioridad puede tomar valores entre 0 (normal) y 7 (prioridad máxima para control de red) aunque en la práctica no sirve de mucho en Internet, ya que gran número de las implementaciones existentes del protocolo suelen hacer caso omiso de este campo.

Los tres bits siguientes son *flags* que reciben el nombre de D, T y R respectivamente. El bit D, *Delay* (retardo), indica una solicitud de bajo retardo; el T, *Throughput* (caudal), indica una solicitud de caudal alto; y el R, *Reliability* (fiabilidad), sirve para indicar una solicitud de fiabilidad. Aunque pueda parecer un mecanismo de especificación de *QoS* (*Quality of Service*, Calidad de Servicio), la red Internet no es capaz de asegurar ninguno de estos parámetros. Sin embargo, sí suelen ser empleados como indicación para la función de encaminaminamiento. Por poner un ejemplo, supóngase que se emplea IP para dar servicio a una aplicación de diálogo interactivo (caso A). Será más importante obtener una rápida respuesta del interlocutor que disponer de la capacidad de transmitir grandes cantidades de información por segundo. Por ello convendrá activar el bit D. Si por el contrario se desea transferir enormes ficheros (caso B), no importará tanto el retardo de la respuesta como la brevedad del tiempo de transferencia, por lo que interesará activar el bit T. Aunque la red no pueda asegurar una calidad de servicio, si en determinado punto de la ruta un nodo intermedio tiene la alternativa de retransmitir, entre otros, por un enlace de satélite (alto retardo, alto ancho de banda), probablemente la

**Un protocolo queda definido por el formato de sus PDUs y el servicio que presta**

función de encaminamiento del protocolo elegirá este enlace para el caso B, pero no para el caso A.

Los dos últimos bits se encuentran siempre a 0, ya que no se les ha asignado semántica. Es la clase de bits que se suele denominar "reservados para uso futuro".

- ***Longitud total del datagrama***: Análogamente a la cabecera, al poder transportarse un cuerpo de datos de longitud variable, es necesario indicar el tamaño del datagrama. Como puede observarse en la figura, este campo tiene una longitud de 16 bits, por lo que el datagrama de tamaño máximo tendrá una longitud de 2 elevado a 16 bits, es decir, 64 kb.
- ***Campos de fragmentación***: Los bytes 5 a 8 de la cabecera del datagrama se emplean para gestionar la fragmentación IP, otra de las funciones más importantes del protocolo, por lo que se estudiarán con más detalle en el apartado siguiente (campos Identificador, *DF*, *MF* y *Offset*)
- ***TTL***: Este es el campo de tiempo de vida (*Time To Live*). El TTL es un valor que se va decrementando en cada salto que da el datagrama. Es decir, al ser recogido en un nodo intermedio, éste decrementa el valor del TTL en una unidad y lo reenvía al siguiente. Si el TTL toma un valor 0, se elimina el datagrama. Esto, como se mencionaba anteriormente, no supone ningún problema para el protocolo, que puede "tirar" datagramas, ya que está especificado como un protocolo no fiable. De esta forma se limita el número máximo de saltos que puede dar una datagrama, lo que supone una protección frente a posibles bucles debido a errores en el encaminamiento.

**Se denomina usuario a aquel programa de aplicación o protocolo que utiliza los servicios de otro protocolo**

- ***Protocolo***: En este campo se indica cuál ha sido el protocolo de nivel superior que ha generado la información del datagrama y que determinará el formato de los datos. Aunque para IP sólo son "mercancía transportada". De esta forma, en recepción, el software IP es capaz de entregar la información al software que implementa el protocolo correspondiente capaz de interpretarlos (ver figura 3).
- ***Checksum***: El campo de *checksum* contiene información redundante con objeto de certificar la integridad de los datos de la cabecera del datagrama.
- ***Campos opcionales***: Se reserva un espacio para campos que transporten información adicional. Con ello se permite experimentar a los programadores de protocolos, así como probar nuevas funciones. Entre los campos opcionales que define la especificación se encuentra la posibilidad de realizar registros de ruta (se van añadiendo los sistemas intermedios por los que ha pasado el datagrama), registros de tiempos (se van añadiendo los tiempos intermedios), indicar en la cabecera la ruta a seguir (que quedaría impuesta desde el origen), etc. El lector que desee profundizar más en este aspecto encontrará toda la información en el RFC.

**El usuario de un protocolo no tiene por qué encontrarse en un nivel superior del modelo de referencia**

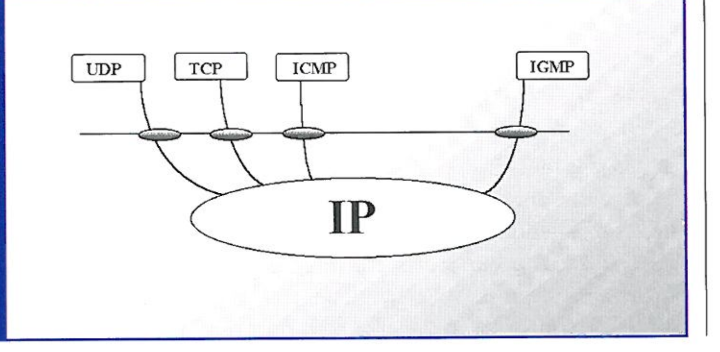



**FIGURA 3:** En función del valor del campo protocolo, IP conoce a qué protocolo va destinada la información que transporta en cada datagrama.

### NECESIDAD DE UNA FRAGMENTACIÓN DE DATAGRAMAS

La fragmentación consiste en la división de un datagrama en dos o más datagramas más pequeños y es una de las principales funciones llevadas a cabo por IP. Para entender por qué se realiza esta función es necesario conocer un concepto que se maneja en el nivel de enlace: el concepto de *MTU*.

Un datagrama IP, como se explicó en el artículo anterior, no viaja "tal cual" por cada una de las redes reales. Al igual que el nivel de red toma la información del nivel de transporte y le añade unas cabeceras propias para formar un datagrama (*PDU* de red), el datagrama es entregado al nivel de enlace donde todo él pasa a ser datos para ese nivel que, a su vez, le añade sus propias cabeceras formando lo que se denomina una *trama* (*PDU* de enlace), figura 4. Es esta trama la que realmente viaja por cada una de las redes (convertida por el nivel físico en señales electromagnéticas) reales que forman la internet virtual. Cada una de estas redes puede contar con una tecnología diferente, por lo que el tamaño máximo de la trama puede variar. Es este tamaño máximo de trama a transmitir lo que recibe el nombre de *Maximum Transferable Unit* o *MTU*. Como ejemplo más común, la *MTU* de una Ethernet es de 1500 bytes, pero existen otras tecnologías con *MTUs* mayores o mucho menores. Como es lógico, si el datagrama es más grande que la *MTU* de la red por la que se quiere transmitir habrá que descartarlo o fragmentarlo,

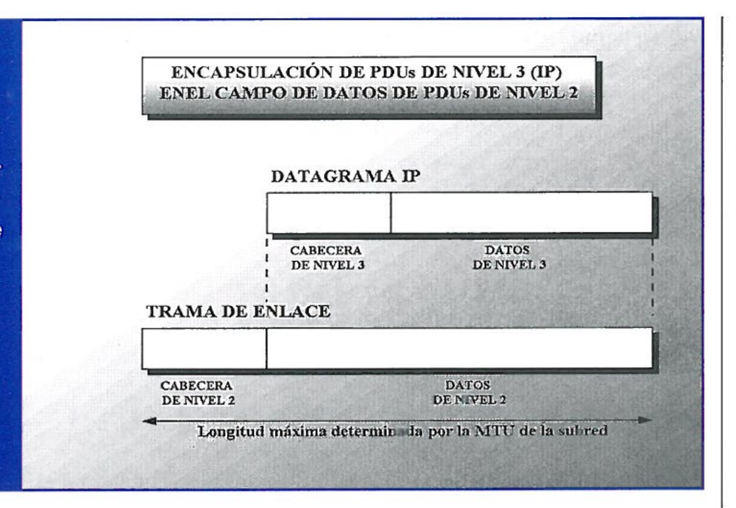



**FIGURA 4: Ejemplo de la encapsulación de PDUs de nivel n+1 en PDUs de nivel n, de donde surge la necesidad de fragmentación.**

por lo que se opta por esta segunda opción. A la hora de diseñar IP existía la alternativa de hacer coincidir la longitud máxima del datagrama IP con la mínima *MTU* que se pudiera encontrar en Internet. Pero esta solución se cae por su propio peso. En primer lugar, supondría un gran desperdicio de recursos en aquellas redes capaces de transmitir tramas mucho mayores. En segundo lugar, no es posible predecir la mínima *MTU* que podría surgir en el futuro y, en tercer lugar, pero no menos importante, la filosofía de IP se basa en la generación de una red virtual transparente, independiente de las redes reales sobre las que se sustenta y que deben quedar ocultas a los niveles superiores. Es por ello por lo que se justifica el mecanismo de fragmentación.

## CONTROL DE FRAGMENTACIÓN IP

A primera vista podría parecer que segmentar es fácil, basta con "partir" el datagrama y re-enviar los trozos. Pero en una red de conmutación de paquetes el problema surgirá a la hora de re-ensamblar los diferentes segmentos. Recuérdese que el IP recibe *SDU's* completas del nivel superior y entrega *SDU's* completas al nivel superior. Por ello es necesario re-ensamblar los datagramas recibidos para reconstruir la *SDU* que se proporcionará al nivel superior. Por ello, las cabeceras IP deben contener la suficiente información de fragmentación que permita una ordenación adecuada de los fragmentos en destino, eliminando los posibles fragmentos duplicados y siendo capaz de recuperar los perdidos.

Así, la entidad origen asigna un valor al campo Identificador (figura 1) de cada datagrama. En caso de que el datagrama se segmente, todos sus segmentos quedarán asociados porque compartirán este campo identificador. No existe posible confusión entre datagramas pertenecientes a diferentes comunicaciones y que contengan el mismo valor en el campo de identificación, ya que los datagramas quedan unívocamente identificados por la pareja de valores (Identificador, Dirección Fuente).

El campo Identificador sirve para determinar qué segmentos pertenecen a un datagrama original, pero no aporta la información necesaria para su posterior re-ensamblado. Así, se emplea un campo *Offset* de 13 bits para indicar la posición del fragmento en el datagrama. Esta posición se expresa en unidades de 8 bytes con objeto de ahorrar espacio a la hora de indicar el *offset*. La razón de que el campo *Offset* sea de 13 bits es que con 13 bits se pueden definir hasta 8192 posibles fragmentos. Como se ha dicho, cada fragmento (salvo el último, en el que se podría "acabar" la información antes) debe tener una longitud múltiplo de 8 bytes. Por tanto, en el peor caso, de máxima fragmentación del datagrama de longitud máxima, se podría disponer de hasta 8192 fragmentos de 8 bytes lo que supone, exactamente, 64 kb, longitud máxima del datagrama IP. Menor número de bits de *Offset* no permitiría una fragmentación máxima y mayor número implicaría que el bit más significativo se encontraría siempre a 0 porque no podría haber tantos fragmentos. Aún así, existen dos bits más de control de fragmentación:

- *DF*: *Don't Fragment* (No fragmentar). Cuando se encuentra a 1, indica que el datagrama no se puede fragmentar, generalmente porque se sabe que la entidad destino no será capaz de re-ensamblarlo. El valor 0 indica datagrama fragmentable.
- *MF*: *More Fragments* (Más fragmentos). Cuando se encuentra a 1 indica que existen más fragmentos del datagrama original. Un valor 0 indica que se trata del último fragmento.

La labor de re-ensamblado de fragmentos se realiza únicamente por la entidad destino. No tendría sentido realizarla en un nodo intermedio ya que no existe ninguna grantía de que todos los datagramas vayan a pasar por ella ya que, al tratarse de una red de conmutación de paquetes, cada datagrama puede recorrer una ruta distinta. Por otra parte, aunque esto no fuera así, también existiría la posiblilidad de una

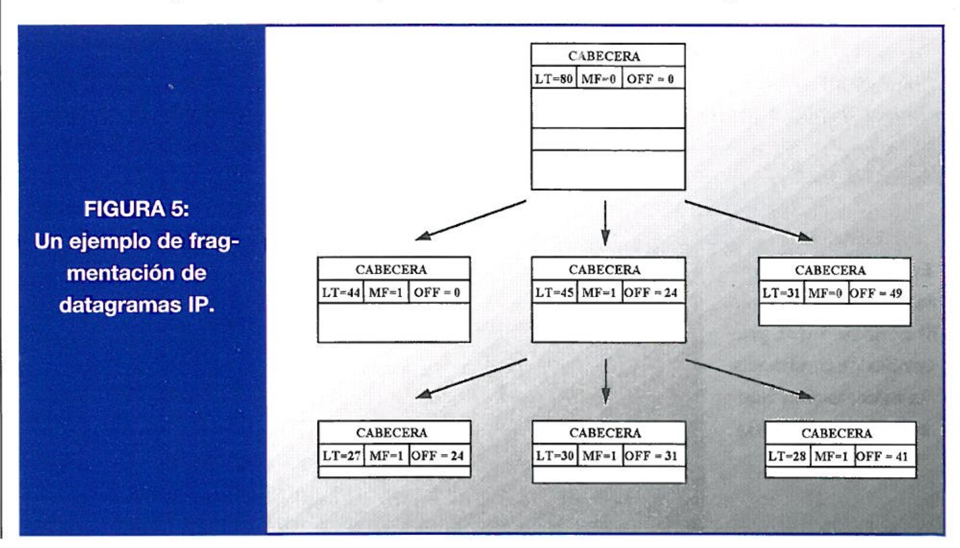



**FIGURA 5: Un ejemplo de fragmentación de datagramas IP.**

nueva fragmentación posterior, con lo que se estaría desperdiciando recusos.

Al recibir un fragmento una entidad IP pone en marcha un temporizador a la espera de completar el datagrama original. Si vencido un *time out* no ha sido capaz de recomponerlo porque no ha recibido todos los fragmentos, lo descarta.

## UN EJEMPLO DE FRAGMENTACIÓN

Con objeto de dejar claro el significado de los campos relacionados con la fragmentación, se ha desarrollado un ejemplo que se ilustra con la figura 5. Para el ejemplo, en que sólo se ha utilizado el campo de longitud total, se considerará que las cabeceras de los datagramas únicamente constan de los campos opcionales, por lo que tienen una longitud de 20 bytes. Como punto de partida se toma un datagrama inicial con una longitud total (en adelante LT) de 80 bytes, por lo que únicamente 60 serán de texto. Lógicamente MF=0, al tratarse del datagrama inicial.

no se llegará jamás a tener noticia de la existencia del datagrama B (a no ser que se haya producido un duplicado). Por ello, los *offsets* deben ser también relativos al datagrama original y encontrarse en un rango comprendido entre 24 y 49, rango que cubría el texto de éste. Operando como en la primera fragmentación se obtienen los *offsets* de 24, 31 y 41. Este método de referencia de la información al datagrama original no sólo es el único que permite reconstruir el mismo sino que, además, resuelve el problema de duplicados, al detectar el software de comunicaciones la superposición de fragmentos, así como la detección de "agujeros" para los que no ha llegado fragmento (pérdidas). En cualquier caso, recuérdese que IP no es fiable, por lo que podrá descartar datagramas enteros, pero nunca aceptará uno incompleto.

Como se introdujo en los primeros apartados, el problema de la fragmentación se encuentra relacionado siempre con cuál será la máxima unidad de

# La fragmentación de datagramas es una de las funciones críticas de IP

En uno de los nodos intermedios, este datagrama debe ser fragmentado en otros tres. Como ejemplo, un primer datagrama, A, tomará los 24 primeros bytes de datos; un segundo, B, los 25 intermedios; y un tercero, C, el resto, es decir, 11. De esta forma, las LT de los datagramas-fragmento, con sus cabeceras, resultan ser de 44, 45 y 31 bytes respectivamente y sus *offsets*, de 0, 24 y 49 respecto al texto original. Los datagramas A y B tendrán valores MF=1, pero C tendrá MF=0 al tratarse del último.

Supóngase ahora que el datagrama B sigue su ruta (se le pierde la pista a los otros dos) y en otro de los nodos intermedios debe también ser fragmentado. En este caso, su longitud de texto, 25 bytes, se reparte, por ejemplo, en fragmentos de 7, 10 y 8 bytes respectivamente. Como todos provienen de un datagrama con MF=1 está claro que el último de los fragmentos de este datagrama no puede ser el último de los fragmentos del datagrama original. Por ello, todos llevaran MF=1. Del mismo modo, en la entidad destino

información que está dispuesto a aceptar el nivel inferior y no se realiza únicamente en el nivel 3. Así, por ejemplo, la *SDU* de tamaño máximo de IP es de 64 Kb. Por ello, si el nivel superior necesita enviar más información a IP, será el nivel superior y no IP el encargado de realizar una segmentación de la misma en *SDU's* de 64 kb. Como se puede apreciar, existe cierto tipo de funciones que se repiten en varios niveles con el objeto de permitir la definición de interfaces estándar entre niveles que permitan reemplazar uno de ellos por otro que se ajuste a la misma interfaz sin que afecte a la comunicación. Esta es una de las características principales de los sistemas abiertos. Sin embargo, implica también la introducción de cierto *overhead* al sistema, uno de los problemas con que se encuentra el modelo OSI, que cuenta con demasiados niveles.

## CONCLUSIONES

Con la excusa del estudio del caso particular más extendido, el protocolo IP, y a la vez que se trata de hacer comprender al lector el funcionamiento de la red Internet, se busca un objetivo más amplio: el estudio de los diversos conceptos asociados a las arquitecturas de protocolos en sistemas abiertos. Así, a la vez que se ha planteado un repaso sobre la idea de descomposición de una comunicación en niveles aplicando la terminología OSI a una arquitectura TCP/IP, se ha introducido los conceptos de fragmentación y *MTU*. En futuros artículos se continuará desarrollando el nivel 3 de Internet (encaminamiento IP e ICMP) y se procederá también al estudio del TCP, que proporciona la capa más alta en Internet que da servicio a las aplicaciones, servicio que en anteriores artículos se han empleado para la programación de comunicaciones.

## BIBLIOGRAFÍA


La especificación completa del protocolo se encuentra en:
[RFC 791] "Internet Protocol", DARPA INTERNET Program Protocol Specification. Information Sciences Institute, University of Southern California. que se facilita con la revista.

Bibliografía complementaria de interés didáctico:
- Comer, Douglas E., "Internetworking with TCP/IP. Volume I: Principles, Protocols, and Architecture". Prentice Hall
- Comer, DOuglas E. & Stevens, David L. "Internetworking with TCP/IP. VOlume II". Prentice Hall
- Tanembaum, Andrew S. "Redes de Ordenadores". Prentice Hall.


## CONTACTAR CON EL AUTOR

Para cualquier duda, comentario, sugerencia o crítica, se anima al lector a que se ponga en contacto con el autor mediante carta al correo del lector o, preferible, por medios electrónicos a:

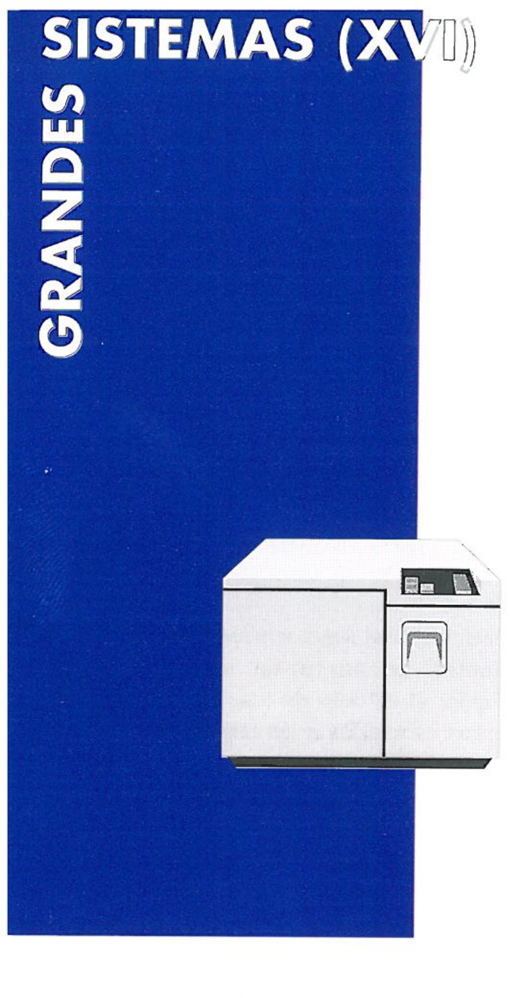

E-mail Internet: echeva@dit.upm.es
E-mail CompuServe: 100646,2456
http://highland.dit.upm.es:8000


## CORRECCIÓN

En el artículo correspondiente al número 25 se comentaba que la igualdad de datagramas se mantenía entre parejas de máquinas conectadas directamente, pero no entre extremos. Si bien esto es cierto dado que campos como el *TTL* varían de salto en salto, el ejemplo que se puso mencionando las direcciones origen y destino es incorrecto, ya que se conservan durante todo el trayecto. El mes próximo se justificará esta idea.

# UTILIDADES MVS (I)

José María Peco



Evidentemente un tema tan amplio como el de las utilidades requiere mucho más que un artículo para poder describir cada una de ellas, razón por la cual sólo se han elegido aquellas que pueden ser más representativas dentro de cada época, y que a su vez son de gran utilidad para cualquier desarrollador de un Centro de Desarrollo/Proceso de Datos (CPD)

Con el fin de enmarcar históricamente el origen de las utilidades, se incluye a continuación una definición del término SISTEMA OPERATIVO, ya que las utilidades no son más que programas incluidos o que acompañan a éste. No tienen una función determinada dentro de cualquier aplicativo, pero en cambio son usadas en la preparación y ejecución de cualquier trabajo para editar, listar, borrar, etc., cualquier fichero.

En cualquier caso, hay que tener en cuenta que los sistemas operativos surgieron acompañando al Hardware para el que se escribieron con el fin de hacer más manejables las máquinas.

**Con este artículo comienza un breve tema dedicado a unas herramientas que por su sencillez y potencia permiten realizar un gran número de funciones a nivel del sistema. Son las típicas utilidades del sistema operativo.**

No obstante, cabe recordar que el S.O. es el soporte lógico del hardware, y en consecuencia, como todo informático conoce bien, un mismo hardware puede soportar distintos S.O.

Definición: Sistema operativo es un conjunto de programas, realizados generalmente por el fabricante del equipo y cuya misión es aumentar el rendimiento total del equipo de proceso de datos. Este rendimiento total se ve influido por los siguientes factores:

- **Rendimiento específico:** volúmen total de trabajo realizado por el sistema en un determinado periodo de tiempo (*THROUGHPUT*)
- **Tiempo de respuesta:** intervalo de tiempo que transcurre desde que un usuario somete un elemento de trabajo al ordenador, y el momento en el que recibe los resultados.
- **Disponibilidad:** Es el grado en que un sistema de proceso de datos está preparado cuando se necesita.

## UTILIDADES

Bajo este calificativo se encuentran aquellos programas que están realizados o no por el fabricante del S.O. y que tienen por misión liberar al usuario de codificar rutinas y códigos para la realización de funciones de uso común, tales como:

- mantener y manipular datos del sistema.
- reorganizar, cambiar, comparar o manipular datos y/o ficheros de usuario.

**Las utilidades no se diferencian en nada de cualquier programa de aplicación**

Las utilidades no se diferencian en nada de cualquier programa de aplicación, a no ser por su finalidad, y pueden ejecutarse como cualquier otro, bien mediante:

- un procedimiento de comandos o CLIST.
- a través de una trabajo *batch* preparando un JCL.

**Figura 1**

**Esquema de paso de JCL para ejecutar una UTILIDAD**

```
//nom_paso EXEC PGM=nom_utilidad
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD ...
//SYSIN    DD *
 fichas de control de la utilidad
/*
//Ddname1 DD ...
//Ddname2 DD ...
//Ddnamen DD ...
```

- Desde cualquier aplicación mediante una llamada al ejecutable correspondiente.

Para la exposición que sigue se va a seguir un JCL que agrupa todas las utilidades que se comentan en el artículo. No obstante, se verán ejemplos aislados con las otras formas de invocarlas.

## ENTORNO DE LA UTILIDAD

Dentro de un JCL, los programas de utilidad necesitan para su control dos tipos de sentencias:

- Sentencias propias de JCL para la asignación de recursos.

**Figura 2**

**Formato del parametro DISP de la ficha DD**

```
                DISP=(anterior,bien,mal)
donde
ANTERIOR puede tomar los valores:
   NEW        si no existía antes
   OLD        si ya existía antes. En este caso los nuevos datos
              sustituyen a los existentes.
   MOD        Ya existe, y los nuevos datos se apendizaran a los
              existentes.
   SHR        Es un fichero compartido.
BIEN y MAL pueden tomar los valores:
   DELETE     si se desea que se borre.
   KEEP       que se mantenga.
   CATLG      que se incluya en el catalogo.
```

- Sentencias de control de la utilidad para especificar a ésta lo que debe ejecutar.

La figura 1 muestra el formato general de cualquier paso de JCL que ejecute un programa de utilidad, donde la *Ddname* SYSPRINT define el destino en el que se escribirán los mensajes devueltos por la utilidad, y la *Ddname* SYSIN contiene las fichas de control de la utilidad, es decir, los comandos que debe ejecutar. El formato de las fichas de control es el siguiente:

*COMANDO operadores comentario*

debiendo resaltarse que cada elemento se encuentra delimitado por blancos, y los distintos operadores separados por comas sin dejar espacios. En el caso de que los distintos parámetros u operadores no quepan en una única línea, deben terminar en coma, dejar al menos un blanco, y escribir un carácter distinto de blanco en la columna 72, considerándose el contenido de la siguiente línea como continuación de los operadores de la anterior.

Así mismo, es de reseñar que todas las utilidades del MVS tienen como característica común el hecho de ubicarse en una librería de uso general, cuyo nombre suele ser SYS1.LINKLIB. Por esta razón debe especificarse la ficha DD que tiene por etiqueta (*Ddname*) STEPLIB para poder incorporar al entorno (alocar) el ejecutable correspondiente.

**Las utilidades no son más que programas incluidos o que acompañan al sistema operativo**

## IEFBR14

Esta utilidad tiene por particularidad que su fuente sólo contiene como instrucción ejecutable 'FIN DE PROGRAMA'. Es decir, por sí mismo no hace nada, pero su ejecución influye en los ficheros asociados en base al parámetro *DISP* que acompaña a cada una de las fichas DD asociadas a dicho paso.

Como ya conoce el lector, este parámetro especifica mediante 3 subparámetros la disposición del fichero al comenzar el paso y la disposición en la que debe quedar si termina bien, y si termina mal, tal y como muestra la figura 2.

Los nombres dados a las fichas DD (de ahí el nombre de *Ddname*) no son significativos en sí mismos, incluso pueden concatenarse especificando una única *Ddname*.

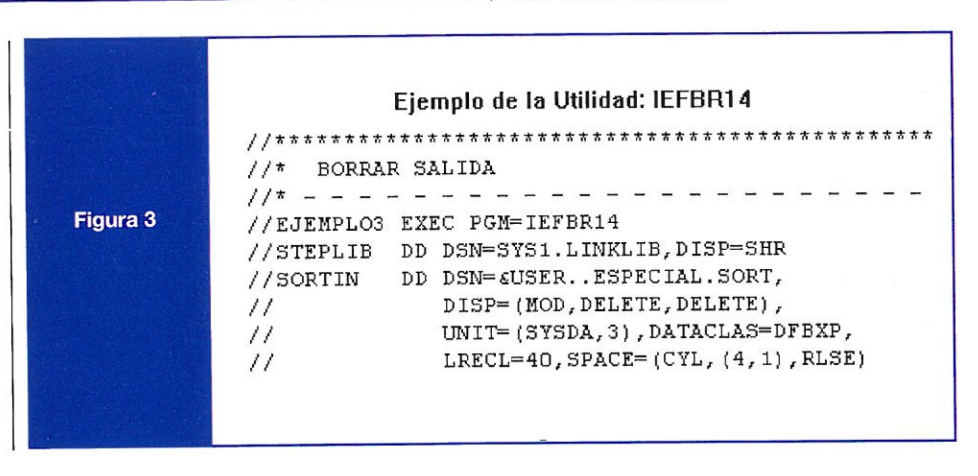


**Figura 3**

**Ejemplo de la Utilidad: IEFBR14**

```
//*********************************************
//*  BORRAR SALIDA
//* - - - - - - - - - - - - - - - - - - - - - -
//EJEMPLO3 EXEC PGM=IEFBR14
//STEPLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//SORTIN   DD DSN=&USER..ESPECIAL.SORT,
//            DISP=(MOD,DELETE,DELETE),
//            UNIT=(SYSDA,3),DATACLAS=DFBXP,
//            LRECL=40,SPACE=(CYL,(4,1),RLSE)
```

**Figura 4**

**Ejemplo de la Utilidad: IEBGENER**

```
//EJEMPLO4  EXEC PGM=IEBGENER
//STEPLIB   DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT  DD DUMMY
//*         ----------- fichero de entrada
//SYSUT1    DD DISP=SHR,DSN=JMPDES.ESPECIAL.REP4
//*         ----------- fichero de salida
//SYSUT2    DD SYSOUT=A,HOLD=YES,DEST=RMT103
//SYSIN     DD DUMMY
```

La figura 3 muestra un ejemplo de uso de esta utilidad. Como ya se verá al comentar el JCL completo que acompaña al artículo, el siguiente paso al mostrado, tiene definido el fichero de salida como nuevo para cuando comience el paso. Por eso en este paso, si existe el fichero cuyo DSN se especifica en la ficha DD de nombre *SORTIN*, lo borra, pues el parámetro *DISP=(MOD,DELETE,DELETE)* especifica que cuando comience la ejecución de este paso, si no existe el fichero, debe crearle, razón por la que se tienen que especificar los parámetros necesarios para crear el fichero; y si ya existía, los datos de salida (que no hay, pues el programa sólo ejecuta un fin de programa) se apendizarían al fichero por tener situación MOD. Cuando termina la ejecución del programa, elige la segunda situación si termina bien, y la tercera si termina mal. Como no hace nada, debe terminar bien, y precisamente se especifica que le borre con el parámetro *DELETE*; y, si termina mal, se especifica que le borre también pues se ha especificado el parámetro *DELETE* en la tercera posición, por lo que en cualquier caso, cuando aloque (asigne al entorno del programa), el fichero, es cierto que no existe, y por tanto debe tener disposición *NEW*.

## IEBGENER

Este programa de utilidad, junto con el anterior, son las utilidades estándar mas usadas en cualquier instalación.

**Los nombres dados a las fichas DD pueden concatenarse especificando una única Ddname**

Básicamente, este programa se limita a copiar el fichero de entrada definido con la *Ddname* SYSUT1 sobre otro de salida definido con la *Ddname* SYSUT2, pudiendo incluso manipular la información contenida en los mismos.

El ejemplo mostrado en la figura 4 realiza la copia de un fichero de entrada sobre la impresora que tiene por nombre lógico *RMT103*, ya que es el destino dado al fichero de salida.

Cuando la *Ddname* SYSIN es *DUMMY* (fantasma) las parámetros *LRECL* y *RECFM* de los ficheros asociados a las *Ddnames* SYSUT1 y SYSUT2 deben ser iguales, pues eso significa que se realiza la copia sin manipular la información.

Por el contrario, cuando se desea manipular la información, debe especificarse dentro del fichero asociado a la *DDname* SYSIN los comandos de la utilidad especificados en la figura 5.

Cuando se desea que el formato del registro de salida (cuando el fichero es nuevo) coincida con el de la entrada, basta con codificar como parámetro *DCB (Definition Control Block)* de SYSUT2 :

*DCB=*.SYSUT1*

En el caso de que los datos de SYSUT1 vengan especificados en la propia cadena (*IN STREAM*), debe especificarse obligatoriamente la *DCB* de SYSUT2.

Cuando la salida se direcciona a la salida del *JOB*, no es necesario especificar *DCB*, ya que asume la estándar.

*SYSUT2 DD SYSOUT=**

Otro ejemplo de esta utilidad sería el de la figura 6, mediante el cual se desea copiar las posiciones 1-30 de entrada a las posiciones 51-80 del registro del fichero de salida, y las posiciones 51-100 a las posiciones 1-50 del registro de salida.

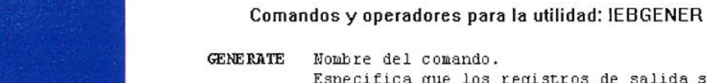
## IEHLIST

Esta utilidad es usada para listar nombres de ficheros.

Como ya se comentó en el primer artículo del tema dedicado a los ficheros de MVS (Solo Programadores-núm.19) La *VTOC (Volume Table of Contains)* es equiparable a la *FAT (File Allocation Table)* del MS-DOS, y contiene la dirección física dentro del volúmen (disco) de todos ficheros contenidos en el mismo.

Por su parte, el catálogo es un archivo que contiene el nombre de los ficheros, y el nombre del volúmen en el que se encuentra.

**Figura 5**



**Comandos y operadores para la utilidad: IEBGENER**

| | |
|---|---|
| **GENERATE** | Nombre del comando. Especifica que los registros de salida son diferentes a los de entrada |
| **MAXFLDS** | Operación de GENERATE. Especifica el numero máximo de elementos **field** del comando **record** |
| **RECORD** | Nombre del comando. Especifica el formato de los registros de salida |
| **FIELD** | =**(parm_1,parm_2,parm_3,parm_4)** siendo<br>**parm_1**: tamaño del área de registro de salida<br>**parm_2**: posición del primer byte seleccionado<br>**parm_3**: posible conversión<br>**parm_4**: posición del primer octeto del área de salida. |

Especifica el formato de los distintos campos de salida. Ejemplo:(30,1,,51) especifica que las 30 primeras posiciones del registro de entrada debe colocarlas a partir de la posición 51 en el registro de salida

Figura 6

```
//EJEMPLO5  EXEC   PGM=IEBGENER
//SYSPRINT  DD     DUMMY
//STEPLIB   DD     DSN=SYS1.LINKLIB,DISP=SHR
//SYSUT1    DD     DSN=JMPDES.ENTRADA,DISP=SHR
//SYSUT2    DD     DSN=JMPDES.SALIDA,DISP=(NEW,CATLG,DELETE)
//                 UNIT=SYSDA,
//                 SPACE=(CYL,(40,10),RLSE),
//                 DCB=BLKSIZE=23440,LRECL=80,RECFM=FB)
//SYSIN     DD     *
            GENERATE MAXFLDS=2
            RECORD FIELD=(30,1,,51),FIELD=(50,51,,1)
/*
```

De esta forma, combinando ambas búsquedas puede localizarse cualquier fichero o *dataset* de la instalación.

También, a modo de recordatorio, se comenta que los ficheros particionados o *PDS (Partitioned Data Set)* se componen de dos partes, el directorio y el área de datos, estando referenciado en la *VTOC* el área de directorio. Por su parte, este área contiene las referencias a cada uno de los miembros incluidos en el área de datos del fichero PDS.

Los comandos admitidos por esta utilidad a través de la SYSIN son:

- *LISTVTOC:* Para listar la *VTOC*.
- *LISTPDS:* Para listar el directorio.
- *LISTCTLG:* Para listar el catálogo.

Como particularidad se puede citar el hecho de que este programa, para poder ejecutarse, debe contener fichas DD para alocar (incorporar al entorno) el volúmen que se referenciará en el comando a ejecutar en la *SYSIN*. La figura 7 contiene un ejemplo de esta utilidad.

Figura 7

**Ejemplo de la utilidad: IEHLIST**

```
//EJEMPLO6 EXEC PGM=IEHLIST
//STEPLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//DISCO1   DD UNIT=SYSDA,VOL=SER=PACK11,DISP=SHR
//DISCO2   DD UNIT=SYSDA,VOL=SER=PACK12,DISP=SHR
//DISCO3   DD UNIT=2314,VOL=SER=PACK13,DISP=SHR
//SYSIN    DD *
  LISTVTOC VOL=SYSDA=PACK11                                 (1)
  LISTVTOC VOL=SYSDA=PACK12,FORMAT,DSN=(JMPDES)             (2)
  LISTVTOC VOL=2314=PACK13,DUMP                             (3)
  LISTPDS  VOL=SYSDA=PACK11,DSN=JMPDES.PRE1                 (4)
  LISTPDS  VOL=SYSDA=PACK12,DSN=JMPDES.PRE2,FORMAT          (5)
  LISTCTLG VOL=SYSDA=PACK11                                 (6)
  LISTCTLG VOL=SYSDA=PACK12,NODE=JMPDES                     (7)
/*
```

## La VTOC es equiparable a la FAT del MS-DOS, y contiene la dirección física de todos los ficheros

A la vista de dicha figura, y teniendo en cuenta que los distintos listados se direccionan a la cola de salida A, los comandos que ejecuta este programa son:

(1) Lista todas las entradas del *VTOC* del volúmen alocado con disco1.

(2) Lista todas las entradas que cuelgan del prefijo (puntero) *JMPDES*, en formato editado.

(3) Lista la *VTOC* del volúmen alocado con la *Ddname* disco3 en formato hexadecimal.

(4) Lista el directorio o relación de miembros del fichero PDS cuyo DSN se especifica en el mismo directorio.

(5) Lo mismo pero en formato editado.

(6) Lista el catálogo que reside en el volúmen alocado con la *Ddname* Disco1.

(7) Lista el catálogo que reside en el volúmen alocado con la *Ddname* disco2, las entradas asociadas al cualificador (puntero) *JMPDES*.

### PRÓXIMO ARTÍCULO

Puesto que el espacio no perdona, se aplaza para el próximo mes las utilidades *IEBCOPY*, *IEBPTPCH* e *IEHPROGM*, así como otros dos programas de suma importancia en cualquier gran instalación.

Los programas *IDCAMS* e *IKJEFT01* o programa *TSO* para ejecutar en batch.

### UTILIDAD

La utilidad que acompaña este mes al artículo tiene por finalidad listar los objetos natural cuyo nombre figura en un fichero que sirve de entrada y que se denomina con la variable &ENTRADA. A su vez, este fichero de entrada se ve incrementado con los objetos referenciados en los objetos listados.

A ser posible, en el próximo número se comentará con más detenimiento esta utilidad, así como las modificaciones introducidas en los programas de la utilidad que acompañaron al tema anterior para adaptarlos a estas nuevas necesidades.

Como curiosidad, este JCL muestra cómo se puede ejecutar de forma iterativa un JCL, pues el ultimo paso del mismo vuelve a enviar a la cola de entrada (*submite*) el mismo JCL.

# SISTEMAS OPERATIVOS DE RED (I)

*María Jesús Recio*

**Existen diferentes sistemas para manejar una red. Algunos de ellos forman un sistema operativo, otros únicamente ofrecen el entorno de manejo de la red, y finalmente existen aplicaciones que incorporan facilidades en la conexión en red.**

Como ya se ha visto en artículos anteriores de esta serie, las comunicaciones a través de red se establecen vía protocolos de comunicaciones, independientemente del modo físico que soporta dicha comunicación (módem, red local, línea de alta velocidad): recuérdese la visión general dada en el artículo anterior sobre los protocolos TCP/IP.

Existe gran variedad de protocolos que permiten la comunicación entre ordenadores. A veces, estos protocolos forman parte del sistema operativo, que crea así sus propios protocolos (como es el caso de Windows 3.11, que incorpora sus propios protocolos "Red de Microsoft"), o bien se pueden encontrar como software independiente (por ejemplo PC/TCP de FTP Software, que es una versión de TCP/IP para DOS), y otras veces los sistemas operativos incluyen como parte de su software protocolos muy difundidos, como es el caso de TCP/IP con UNIX.

Inicialmente, cada sistema ofrecía de forma exclusiva sus protocolos de comunicaciones. En la actualidad, y debido a la necesidad de interconexión entre sistemas, la mayoría ofrece, además de sus protocolos básicos, otros protocolos de comunicaciones. Por ejemplo, aunque originalmente Netware de Novell trabajaba con protocolos IPX/SPX de manera exclusiva, hoy ofrece módulos para trabajar con otros protocolos como son TCP/IP, protocolos en entorno SNA *(System Network Arquitecture)* de IBM que posibilitan la conexión con grandes sistemas, e incluso protocolos de entorno Macintosh, además de protocolos normalizados OSI. Windows 95 es otro ejemplo claro: ofrece la posibilidad de instalar numerosos protocolos de comunicaciones según las necesidades de la máquina, en cuanto a conexiones se refiere.

La primera consideración que uno debe hacerse al hablar de sistema operativo de red es pensar a qué se llama sistema operativo de red. En la bibliografía publicada en relación a este tema existen algunas discrepancias: analizandolo de forma muy básica, mientras que para algunos un sistema operativo de red es aquel que está concebido como tal, y que se instala de forma que exista una máquina (servidor utilizándose o bien en un sistema centalizado o bien en un sistema distribuido), para otros un sistema operativo de red es un sistema que incorpora funciones para la compartición de recursos entre máquinas. Para los primeros no sería un sistema operativo de red Windows para Trabajo en Grupo mientras que para los segundos si.

Existen otras opiniones que no consideran un sistema operativo de red aquel que no ofrece herramientas para la gestión y control de errores. Y la cosa no se queda aqui, sino que muchos consideran que no es red un conjunto de ordenadores conectados entre sí y ejecutando un emulador de terminal, por ejemplo una sesión telnet, contra una máquina Unix. En el artículo se va a tratar como sistema operativo de red a todo aquel que ofrezca la posibilidad de compartir recursos con otros ordenadores de la red.

Teniendo en cuenta esto, se debe distinguir entre lo que es un sistema en el que todas las máquinas tienen el mismo protagonismo y se conectan unas a otras únicamente para la compartición de recursos, sistemas que reciben el nombre de "sistemas operativos para redes de igual a igual" de sistemas en los que existe una estructura

en la que se tienen servidores de diferentes naturaleza y en las que la comunicación se establece únicamente entre las distintas máquinas y los servidores, no existiendo la posibilidad de ver máquinas de la red que no estén definidas como servidores, y que se denominan sistemas operativos basados en servidor, concepto que debe distinguirse del de arquitectura cliente-servidor.

Los sistemas operativos de red pueden seguir diferentes arquitecturas en su implementación al igual que las aplicaciones. Se pueden encontrar sistemas centralizados en los que todo el procesamiento se realiza en una máquina, en contraposición a los sistemas distribuidos en los que el trabajo se comparte entre diferentes máquinas. Una forma de compartir el trabajo es siguiendo una política totalmente descentralizada y distribuida y otra es siguiendo una política cliente-servidor.

**Figura 1: Relación cliente - servidor**

**Existen dos tipos básicos de sistemas operativos de red: *peer to peer* y orientados a servidor**

## ARQUITECTURA CLIENTE/SERVIDOR

La arquitectura cliente-servidor consiste en la compartición de la carga de procesamiento entre varias máquinas. Se puede decir que se trata de una arquitectura que utiliza los recursos de todos los ordenadores, en contraposición a una arquitectura centralizada en la que una máquina anfitrión realiza todas las operaciones enviando los resultados al resto de las máquinas. Del mismo modo, en este tipo de arquitectura se comparte el trabajo entre las estaciones sin llegar a ser una arquitectura totalmente distribuida en la que cada máquina realiza su trabajo y comparte algunos recursos.

A la hora de hablar de una arquitectura cliente-servidor se debe tener muy presente el entorno en el que se explica. Aunque este concepto hace referencia a la repartición del trabajo, no es lo mismo hablar de un sistema operativo con arquitectura cliente-servidor que de una aplicación o que de unos protocolos de comunicaciones. Mientras que la materialización de esta arquitectura es obvia en el entorno de las aplicaciones y de protocolos, no lo es tanto en el entorno de los sistemas operativos.

Un sistema operativo de red implementa esta arquitectura implantando en la red máquinas servidoras y máquinas clientes. En las estaciones clientes se instala el software necesario para dirigir las peticiones hacia el servidor correspondiente cuando sea necesario, o bien dejarlas para que el sistema operativo local las procese. Una aplicación con esta arquitectura va más alla; instala una parte de la aplicación en la estación servidora y otra en los clientes, de modo que las peticiones realizadas por la aplicación no se procesan de forma única en una de las máquinas, sino que el trabajo se reparte entre varias. Para plasmar esto véase el siguiente ejemplo:

Sobre un sistema operativo que implemente esta arquitectura, por ejemplo Netware, se implanta un gestor de base de datos que no siga la arquitectura. La aplicación y las bases de datos se instalarán sobre el servidor de ficheros, y los clientes, cuando quieran acceder a dicha aplicación, enviaran sus peticiones al servidor. Este les mandará la aplicación, y despues la base de datos completa con la que el usuario desea trabajar. Frente a este entorno, se puede poner el siguiente: un sistema operativo con la arquitectura cliente-servidor y una aplicación (el gestor de bases de datos) que tambien la implementa. Ahora en los clientes de debe instalar un módulo del gestor. Cuando una máquina cliente realiza una petición sobre la base de datos, la máquina servidora selecciona los registros solicitados y se los envía al cliente (únicamente estos registros y no toda la base de datos, como en el caso anterior).

En el caso de los protocolos, de igual forma se tiene para cada aplicación que ofrece el protocolo una parte cliente y una parte servidora. Por ejemplo, la familia de aplicaciones ofrecidas de protocolos TCP/IP están pensadas con la filosofía cliente-servidor, de forma que cuando se hace FTP *(File Transfer Protocol*, recuérdese que se trata de un protocolo que permite la transferencia de ficheros entre máquinas) desde una máquina hacia otra es necesario que la máquina contra la que se ejecuta esta orden esté ejecutando un servidor de FTP.

En este tipo de arquitectura, cuando se transporta al mundo de las redes, se encuentran dos tipos lógicos de ordenadores: sistemas frontales *(front-end)*, que son los ordenadores clientes, y sistemas posteriores *(back-end)*, que son los que proporcionan servicios (servidores), de diferentes naturalezas. Por

| TABLA 1 |
| --- |
| **VENTAJAS** |
| La carga de trabajo asociada a las aplicaciones se divide entre todos los ordenadores. Los clientes realizan parte del procesamiento y los servidores el resto. |
| Un porcentaje importante de información se ubica directamente en el servidor y no en las estaciones de trabajo. |
| El tráfico de la red se reduce, ya que el servidor envía al cliente unicamente la información solicitada. |
| La seguridad de los datos es mayor cuando su ubicación es única. |
| Favorece el procesamiento paralelo múltiple |

**Ventajas de la arquitectura cliente-servidor.**

tanto se puede decir que se denomina relación cliente-servidor a la interacción entre los sistemas frontales y los posteriores.

Con este tipo de filosofía los sistemas clientes disponen de toda su capacidad de procesamiento, y ademas pueden utilizar los servicios ofrecidos por los sistemas servidores. Se puede decir que ésta filososfía es la opuesta al modelo centralizado, en el que los usuarios situados en terminales no inteligentes se comunican con ordenadores anfitriones. En este caso, todo el proceso tiene lugar en el anfitrión, y los usuarios de terminales únicamente escriben órdenes que envían a dicho anfitrión y observan el resultado en su monitor. La tabla 1 plasma algunas de las ventajas de esta arquitectura.

Esta arquitectura funciona de la siguiente forma: el cliente y el servidor internamente se dividen en varios procesos (ver figura 2). En un cliente, cuando se recibe una petición desde la aplicación que se está ejecutando, el software de redirección se encarga de enviarla al sistema operativo local o al servidor correspondiente, dependiendo de hacia quién vaya dirigida la petición. En función del sistema operativo y/o de la aplicación, existen variaciones en la cantidad de trabajo que realiza el servidor: a veces el servidor realiza el menor trabajo posible para así poder optimizar sus prestaciones hacia un grupo de clientes cada vez mayor. Sin embargo, en otras ocasiones el servidor trabaja con toda su potencia y gestiona la mayor parte de la carga de procesamiento.

## El sistema para redes *peer to peer* más difundido es Windows para Trabajo en Grupo

En una arquitectura cliente-servidor el procesamiento de la información se divide entre varias máquinas: no debe pensarse que el servidor es únicamente un servidor de ficheros, y por tanto sólo una extensión del disco duro de la máquina cliente. Aunque la aplicación se ejecute en la máquina cliente, cuando se realizan peticiones sobre ficheros localizados en la máquina servidora, ésta es capaz de extraer del fichero global la información solicitada y enviar únicamente ésta. La figura 3 muestra un caso de arquitectura cliente-servidor con una aplicación de bases de datos, en la que el servidor de aplicaciones está separado físicamente del servidor de bases de datos (ficheros de información).

Un ejemplo de sistema operativo de red que adopta esta filosofía es Netware de Novell.

La tabla 2 muestra los requerimientos físicos en un entorno de LAN para el buen funcionamiento de una arquitectura cliente-servidor.

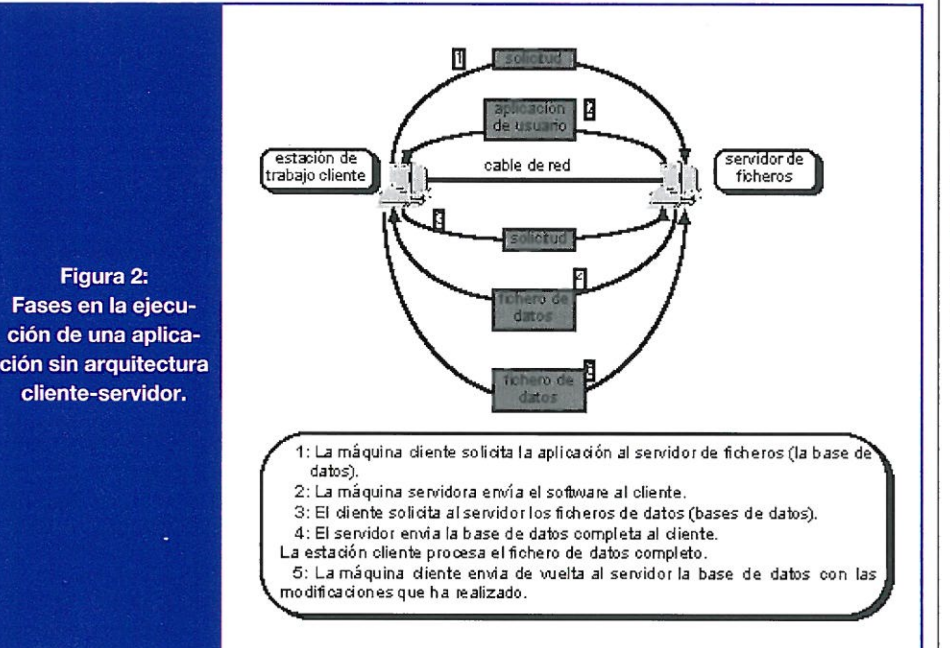
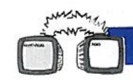



**Figura 2: Fases en la ejecución de una aplicación sin arquitectura cliente-servidor.**

| TABLA 2 |
| --- |
| Discos duros de alta velocidad en los servidores. |
| Velocidad de proceso alta en toda la red. |
| Elevada cantidad de memoria RAM en el servidor. |

**Hardware para los sistemas con arquitectura cliente-servidor.**

A la hora de implantar esta arquitectura se debe tener en cuenta no tanto la topología de la red, como los protocolos de comunicaciones que la van a soportar. Los protocolos existentes iniciales no estaban preparados para las comunicaciones entre aplicaciones, más bien perseguían la velocidad en la transmi-





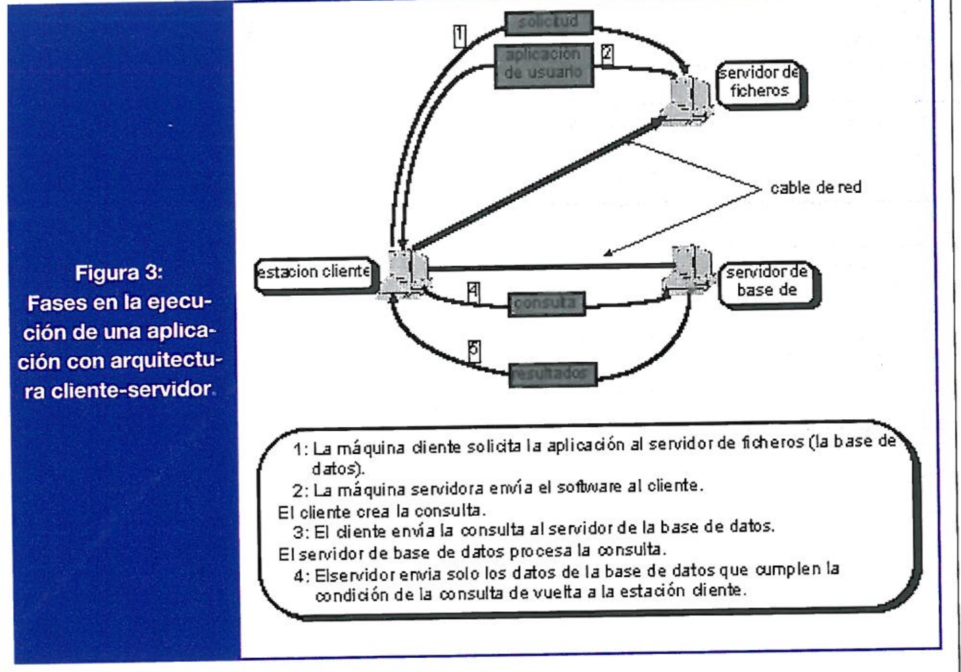



**Figura 3:** Fases en la ejecución de una aplicación con arquitectura cliente-servidor.

sión. Por esto, muchas casas desarrolladoras de software con esta filosofía están creando sus propios protocolos de comunicaciones.

**Los sistemas cliente-servidor con más presencia en el mercado son Netware de Novell, Lan Manager de Microsoft y VINES de Banyan**

### SISTEMAS OPERATIVOS PARA REDES DE IGUAL A IGUAL

Los sistemas operativos para redes de igual a igual *(peer to peer)* se basan en la idea de que todos los ordenadores que constituyen la red comparten sus recursos con el resto, de forma que no existe ninguna máquina especial que gestione la red. Ejemplos de sistemas operativos que siguen esta filosofía son Windows para Trabajo en Grupo (WTG), Netware Lite, LANtastic de Artisoft o NFS. Por ejemplo, cuando tenemos una red en la que todas las estaciones de trabajo tienen WTG, cada una de ellas decide si quiere compartir sus recursos o no (disco duro, impresoras conectadas...). No existe una máquina servidora, cada estación es pontencialmente cliente y servidora a la vez, y además todas las estaciones pueden verse entre sí, siempre que ellas decidan permitirlo.

Esta filosofía es buena cuando la red es pequeña, (no tiene un número elevado de estaciones), y la puesta en común de recursos se realiza de manera ocasional. En cuanto la compartición de recursos se hace elevada empiezan a surgir los problemas, puesto que el sistema no está preparado para soportar esto.

Este tipo de sistemas operativos de red no ofrecen sistemas de seguridad robustos (en algunos casos ni siquiera ofrecen sistemas de seguridad), herramientas para la detección y corrección de errores, aplicaciones para realizar estadísticas de rendimiento ni otras muchas utilidades que se hacen necesarias cuando se trabaja en serio con una red.

**El correo electrónico es una utilidad que la mayoría de los sistemas operativos de red ofrecen**

La mayoría de estos sistemas operativos de red, trabajando en entornos de PC's, lo hacen sobre MS-DOS, por lo que arrastran de alguna manera las limitaciones de este sistema operativo.

Por otra parte, se tiene que reconocer que son muy económicas, fáciles de implementar y muy sencillas de instalar (no requieren la presencia de ningún especialista).

### SISTEMAS OPERATIVOS BASADOS EN SERVIDOR

Se trata de sistemas operativos de red en los que existe una máquina o máquinas que tienen un protagonismo especial dentro de la red. Estas máquinas dentro de la red (servidores) ofrecen sus servicios al resto de los ordenadores de la red (estaciones cliente). Nótese que en este caso las estaciones clientes no se ven entre ellas, sólo ven a aquellas definidas como servidores.

Los servidores implantados en una red con esta estructura pueden ser de diferentes naturalezas: servidores de ficheros (ordenadores que ofrecen la información en ellos contenida al resto), servidores de impresoras (ordenadores que ofrecen servicios de impresión), servidores de comunicaciones (ordenadores que ofrecen habitualmente servicios de fax-módem), y por qué no hablar en entorno Internet, en el que se pueden encontrar servidores de FTP, de páginas web, servidores de dominios, etc...

Los sistemas operativos de este tipo con más presencia en el mercado son Netware de Novell, Lan Manager de Microsoft y VINES de Banyan.

Todos estos sistemas operativos utilizan, desde un punto de vista funcional, el mismo software en las estaciones cliente, que permiten la comunicación con las estaciones servidoras, consiguiendo así la demanda de peticiones de servicio y la respuesta a las mismas.

A diferencia de las redes de igual a igual, las redes basadas en servidor ofrecen, habitualmente, un mayor rendimiento, y pueden atender a un gran número de usuarios, ya que los servido-

res se dedican exclusivamente a satisfacer las peticiones de los usuarios ofreciendo, además, una gran cantidad de accesorios y herramientas de gestión.

En la tabla 3 se pueden ver reflejadas de manera esquemática las diferencias básicas entre las dos filosofías planteadas para sistemas operativos de red.

Las principales características de estos sistemas operativos son:

• **Tolerancia a fallos:** consiste en métodos que aseguren la integridad de los datos y el funcionamiento de la red frente a imprevistos. En la mayoría de las ocasiones consiste en duplicar recursos como pueden ser el disco duro e incluso el servidor entero, de forma que cuando un servidor cae, automáticamente otro entra en funcionamiento, y de esta forma la red sigue estando operativa. De la misma forma, si los datos en un disco duro se deterioran, como se mantiene una copia en tiempo real del disco en otro disco (filosofía de sistemas espejo), bastaría con empezar a trabajar con el otro disco. Dependiendo de las necesidades de la empresa, podrían llegar a duplicarse otros elementos hardware.

• **Aplicaciones basadas en el servidor:** En las redes de igual a igual se pueden compartir recursos, pero de forma limitada: no es posible ejecutar aplicaciones localizadas en otras máquinas. Sin embargo, en la mayoría de los sistemas operativos de red basados en servidor lo habitual es instalar aplicaciones en el servidor de ficheros y ejecutarlas desde las estaciones cliente. Esto ofrece muchas ventajas: rapidez en la instalación (sólo se debe instalar la aplicación una vez), menor coste (solo se necesita un paquete de la aplicación y un número determinado de licencias), menor ocupación de espacio en disco.

Dependiendo del sistema operativo del que se disponga así como de la aplicación, se necesitarán versiones especiales para red, o no.

• **Sistemas de seguridad:** la mayoría de estos sistemas ofrecen sistemas de seguridad elevados, de forma que es necesario identificarse frente al sistema a través de un *login* (nombre de cuenta) y una *password* asociada, para poder entrar en él, y aún así, y dependiendo de las propiedades y privilegios que el administrador del sistema haya dado, se verá limitado el acceso a la información así como el modo de acceder a ella (sólo lectura, permiso de escritura, modificación, creación, borrado, etc.).

A la hora de seleccionar un sistema operativo de red éste es un punto que debe tenerse muy en cuenta, eligiendo un sistema que cumpliera alguno de los criterios de seguridad informática que emiten diferentes organizaciones y gobiernos. Uno de los más extendidos es el criterio de evaluación de sistemas informáticos de confianza del Departamento de Defensa de los Estados Unidos, que establece varios niveles de seguridad que van desde el nivel D (menos riguroso), hasta el nivel A (el más riguroso).

• **Administración de la red:** la imagen del administrador de la red toma una vital importancia en estos sistemas. Por ello se le ofrecen herramientas que le permiten conocer qué usuarios están utilizando la red en un momento determinado, cuáles son las aplicaciones que manejan, cuál es la carga de trabajo, el rendimiento de la red, y todos aquellos elementos que puedan ser de interés para mejorar el rendimiento de la red.

• **Utilidades de diagnóstico:** Algunos sistemas operativos de red dan al administrador algunas utilidades que éste puede utilizar para detectar problemas en la red y para configurarla de forma que opere de manera óptima. Estas utilidades van a aportar datos como son cuellos de botella, caída de máquinas, bloqueos, así como posibles caminos que el administrador de la red puede llevar a cabo para obtener la solución al problema.

• **Correo electrónico:** Se trata de una herramienta muy útil en redes de área local, pues permite la comunicación entre usuarios de forma limpia y sencilla. Un buen sistema de correo electrónico permite almacenar y dirigir mensajes.

Debe señalarse que la mayoría de los sistemas operativos de red de igual a igual también incluyen correo electrónico.

## CP/IP está implementado basándose en una arquitectura cliente-servidor

**TABLA 3**

| CARACTERÍSTICAS | S. O. DE IGUAL A IGUAL | S. O. CLIENTE-SERVIDOR |
|---|---|---|
| Fácil de instalar | SI | NO |
| Fácil de mantener | SI | NO |
| Herramientas para la detección de errores | NO | SI |
| Sistema de seguridad | NO | SI |
| Número alto de usuarios | NO | SI |
| Tolerancia a fallos | NO | SI |
| Altas prestaciones | NO | SI |
| Coste elevado | NO | SI |
| Posibilidad de ejecutar aplicaciones de otras máquinas | NO | SI |
| Distribución del trabajo | SI | SI |

**Principales diferencias entre filosofías de sistemas operativos de red.**

### PRÓXIMO NÚMERO

En el próximo número se hablará de forma más concreta de diferentes sistemas operativos de red pertenecientes a las diferentes filosofías explicadas en esta entrega. El artículo se centrará en los sistemas mas difundidos en la actualidad en el entorno de los PC's.

# EUSKAL PARTY IV

Pedro Antón Alonso

Como cabía esperar el primer día todos nos dedicamos a montar nuestros equipos y a saludar a los amigos, conocidos y desconocidos. Aunque debido a que la festividad del Apóstol Santiago no era tal en todas las comunidades de nuestro país, muchos de los asistentes no llegaron hasta el Viernes por la noche. El resultado total fue de unos 600 asistentes, la mayoría de ellos con su ordenador y mas de 1200 visitantes. No sé si el lector podrá imaginarse a mas de 600 personas apasionadas del ordenador y del mundo de la "demoscene", durmiendo por los suelos del frontón de Caramelo Balda, (lugar donde se celebró la party) unas cuantas horas, antes de continuar programando, componiendo, dibujando, o jugando.

En fin, todo un espectáculo al que merece la pena asistir.

## EL LUGAR DE LA PARTY

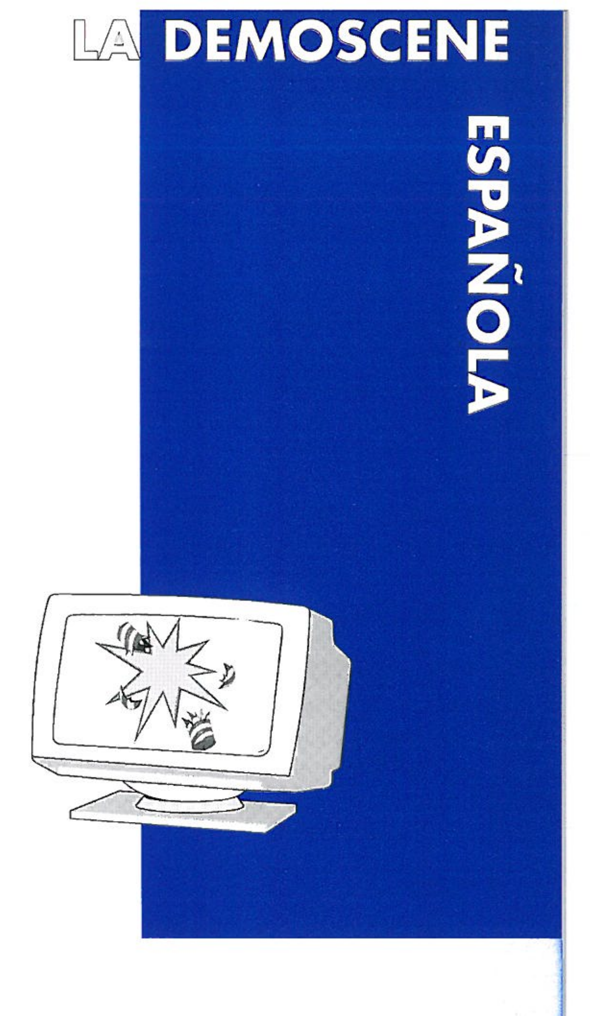
La Euskal Party IV, se celebró en el Frontón de Caramelo Balda, sito en la ciudad de San Sebastián, junto al polideportivo de Anoeta. Grandes medios, un gran despliegue, una buena organización y muchas ganas, por parte de todos, hicieron de esta concentración unos días inolvidables para todos nosotros. Como se puede observar en las fotografías, los ordenadores estaban en el campo de juego, y la organización puso a disposición de los asistentes, toda la infraestructura del frontón, léase duchas, aseos, bar, etc. La parte superior de las gradas del frontón fue empleada por los asistentes para intentar dormir un rato, aunque no faltó quien se durmiera sobre su teclado. No obstante, dormir era más una necesidad biológica que otra cosa, no hubo ni un solo instante en el que no hubiera gente al frente de sus máquinas. La noche se aprovechaba para charlar, ver trabajos de otra gente o mostrar los tuyos propios.

El polideportivo se encontraba limpísimo, y la organización se encargó de que así siguiera a lo largo de toda la party. Había gente (parte de la organización), que se encargaba de limpiar el recinto cada cierto tiempo.

## LA ORGANIZACIÓN

La organización: Sobresaliente.

Como el lector habrá intuido, nada malo se puede decir de la organización. Fue, ante todo, profesional. Se solucionaron los problemas técnicos como se pudo, y todo salió de la manera esperada. Los organizadores se encargaron de buscar patrocinadores, entre los que hay que destacar, Miro, Telefónica, Creative Labs y Jet Internet. Asistiendo además gente de Amigatek, Icon Multimedia, Digital



**Los días 25, 26 27 y 28 de Julio se celebró en San Sebastián la mayor "*party*" de la *demoscene* española, la "*Euskal Party IV*". Sólo Programadores se desplazo a la ciudad donostiarra donde se comprobó que la Euskal comienza a ser toda una institución dentro del periodo estival para todos los amantes de la programación a bajo nivel**

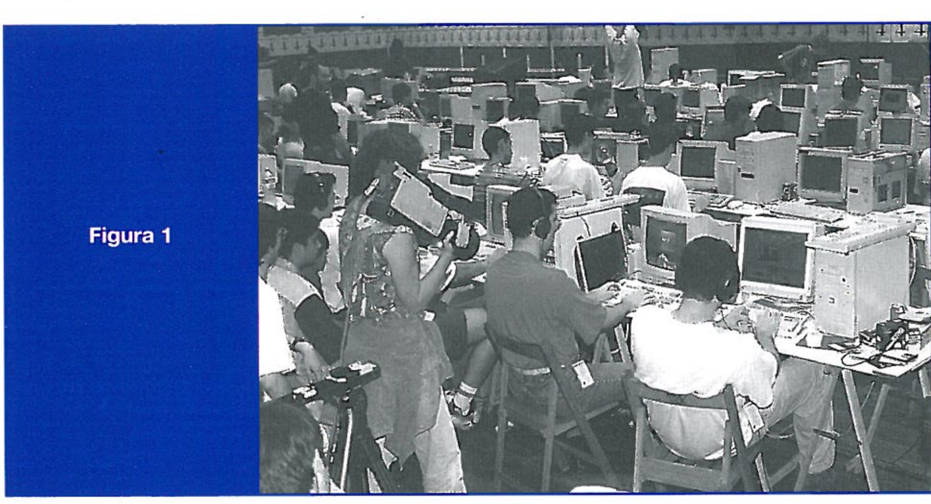


Figura 1

World, Amiga Studio, Escuela de Cine y Vídeo, etc, etc. Destacar la actitud de apoyo a la "demoscene" (algo, por otra parte, totalmente inusual hasta ahora) por parte de Creative Labs, quien regalo 200 joysticks a los asistentes, así como numerosos premios a los ganadores de las distintas categorías, interesándose por los miembros de la demoscene y sus proyectos. En contraposición Jet Internet, que supuestamente debía de proporcionar conexión a Internet a más de 100 ordenadores, sólo fue capaz de dar dicha conexión a poco mas de media docena de ellos.

## LOS ASISTENTES

Prácticamente la totalidad de la "demoscene" española se encontraba en la party.

A un acontecimiento de tales características no pueden dejar de asistir los mejores grupos de la demoscene española, encontrándose en el recinto, miembros de casi todos los grupos españoles. Juan Carlos Arevalo de iGUANA, aprovechó sus vacaciones, para venir desde Dinamarca a la Euskal Party. No faltaron, tampoco, el resto de los miembros del este afamado grupo. La gente de Spanish Lords y Miracle, entre otros grupos, trabajaron duro para terminar sus trabajos antes de la "*deadline*", desafortunadamente no pudieron acabar a tiempo.

Entre los asistentes también se encontraba Luis Crespo, conocido por todos los lectores de Sólo Programadores. Destacar también la presencia de Darkness, miembro de la famosa revista electrónica Imphobia. Y como no, en la party también había sitio para las chicas, que aunque escasas, recibieron saludos en varias de las producciones que se presentaron a concurso, bajo el nombre de *Mermaids*.

## EL CONCURSO

En una party existen distintas categorías de competición.

En la Euskal Party IV existieron todas las categorías de una gran party, y alguna más.

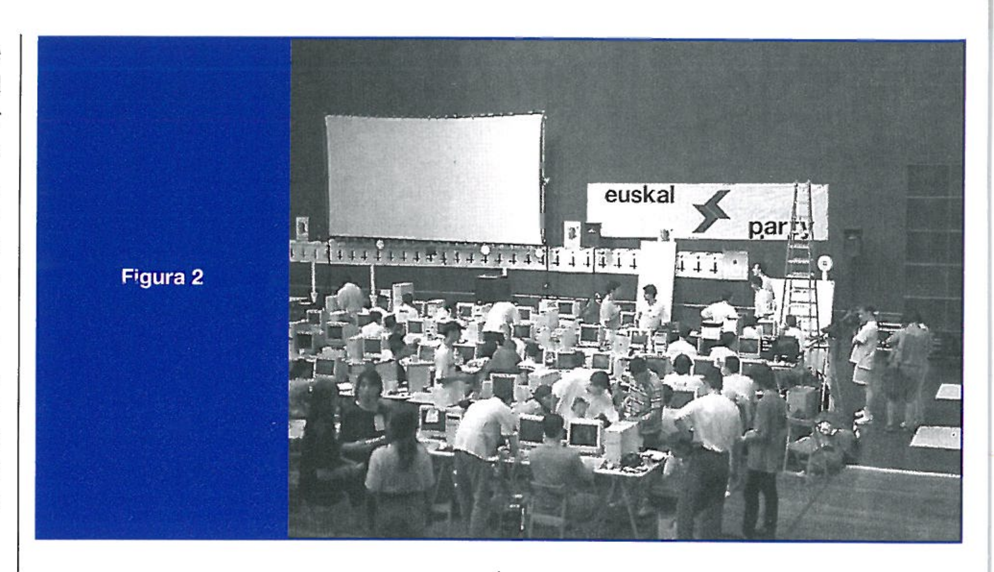


Figura 2

- *Intros de 4K.* El nombre lo dice todo. El tamaño máximo de la producción es de 4K, y no puede tener música.
- *Demos.* La categoría reina de la demoscene. El tamaño máximo de la demo estaba fijado en 4Megabites.

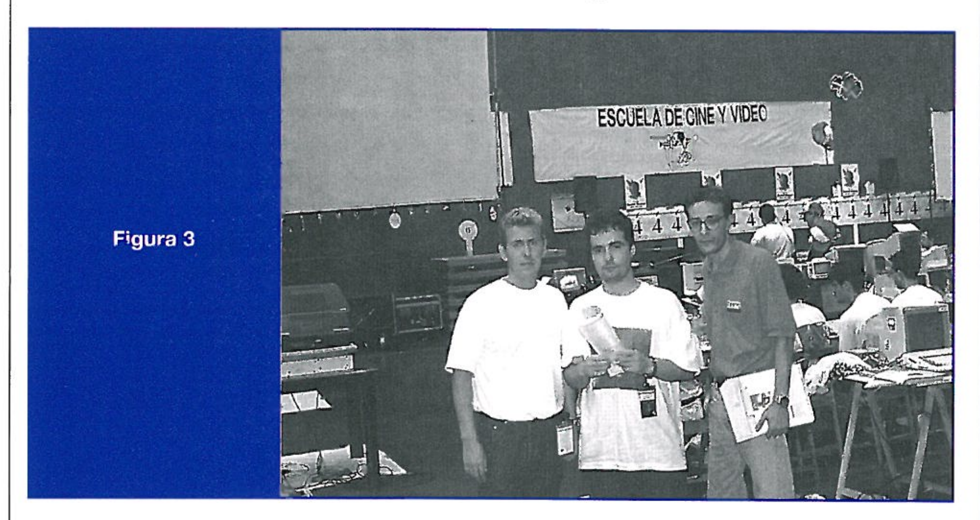
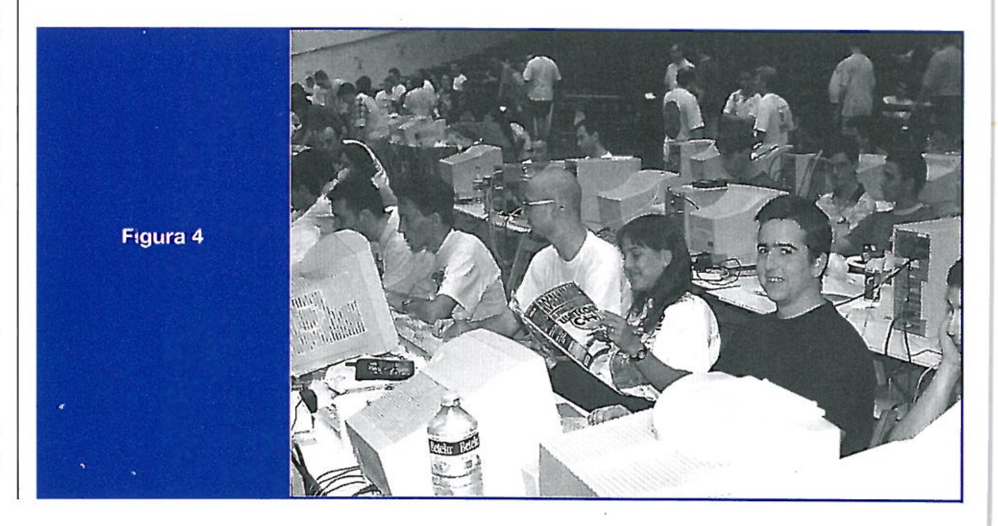


Figura 3

- *Intros de 64K.* El tamaño máximo de estas producciones es de 64K. En esta categoría sí esta permitido incluir música.

Como la party era tanto de Amiga como de PCs, todas estas sus categorías, se dividían, a su vez en Amiga y PCs.


Figura 4

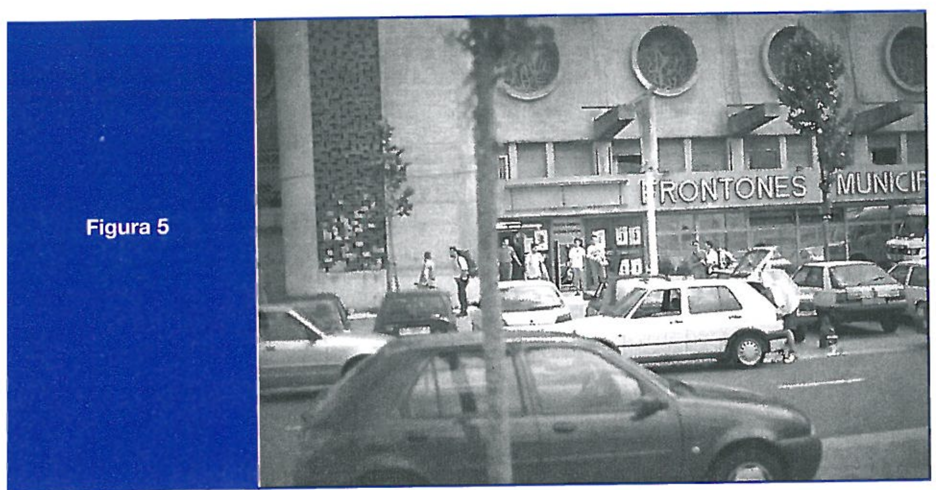



Figura 5

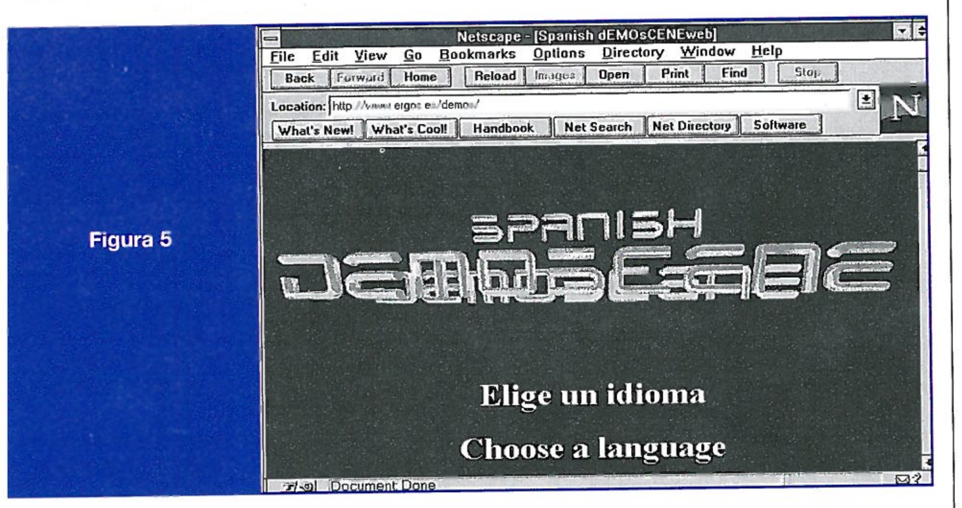



Figura 5

- *Fast 2D.* Los participantes tienen un tiempo máximo para crear un dibujo en 2 dimensiones, basándose en una idea que la organización impone.

- *Fast music.* Los participantes tienen un tiempo máximo para versionar una melodía elegida por la organización.

Los trabajos de estas dos últimas categorías, como supondrá el lector, se crean en la propia party.

- *Gráficos 2D.* Los gráficos que se presenten a esta categoría no deben haber sido diseñados con ningún programa de diseño tridimensional.

- *Gráficos 3D.* Dibujos creados con programas de diseño tridimensional.

- *Módulos de 4 canales.* Las canciones de esta categoría deben tener un máximo de cuatro de canales.

- *Módulos multicanal.* Las canciones de esta categoría pueden tener tantos canales como el autor crea oportuno.

- *Human Demo.* Esta categoría es exclusiva de esta party. En ninguna otra gran party se celebra este tipo de competición. Sin duda alguna, es la mas divertida. El grupo participante debe grabar en vídeo secuencias con efectos similares a los de una demo, pero sin usar el ordenador. Fue realmente divertido ver los trabajos presentados a esta categoría.

Además de todos estos concursos, existió (no podía faltar), la competición de DOOM y de DUKE NUKEM 3D. Algo que habitualmente crea tanta expectación como la propia competición de demos. Así como lanzamiento de disquetes, partido de fútbol Amiga-PC, etc etc.

## LOS TRABAJOS

Muchos y buenos trabajos se presentaron a esta party.

En el recuadro de texto adjunto se pueden observar los trabajos ganadores de esta party, todos ellos estaban hechos con mucha ilusión y gran esfuerzo. Destacar el gran nivel de los trabajos presentados, así como el montaje de la demo ganadora de PC, DESPAIR de iGUANA, en la propia party. Una vez más este grupo nos ha impresionado a todos.

Y esto es todo. Desde estas líneas, animar a los lectores a asistir a la Euskal Party V, la cual será, sin duda, todavía mejor que esta.

Aquellos de vosotros que poseáis acceso a Internet podréis conseguir los trabajos presentados, así como acceder a las páginas de algunos de los grupos de demos de la *demoscene* española en http://www.ergos.es/demos.

## CLASIFICACIÓN DE LA EUSKAL PARTY

### Demos (Amiga)

| | |
|---|---|
| 1: UNDER THE HAMMER por ZABORRA : | 755 |
| 2: SCREAM por AMIGA CIRCLE : | 750 |
| 3: NO MORE LIES por NETWORK : | 480 |
| 4: FIRST por CAPSULE : | 165 |

### Intros de 64K (Amiga)

| | |
|---|---|
| 1: LA HORTALIZA ADVEN. por GOBLINS : | 25 |

### Intros de 4K (Amiga)

| | |
|---|---|
| 1: MD4I por SAPPHIRE / CENTOLOS : | 860 |
| 2: UNAMUNO SÓLO TENIA. por PHORNEE / GOBLINS : | 770 |
| 3: SANCHO PANZA CON... por ASIMMOFF/NEXUS TEAM : | 370 |
| 4: NORK por WIND / NETWORK : | 215 |
| 5: BANDERA A BOLAS por ICEVAN / NEXUS TEAM : | 160 |
| 6: INTRO 4K MÚSICA por DRAREG / OZONE : | 105 |

### Graficos 2D

| | |
|---|---|
| 1: CLINT EASTWOOD por PHORNEE / GOBLINS : | 780 |
| 2: BORRAXO PERDIO por LENTIUM / GOBLINS : | 510 |
| 3: GUN WOMAN por BLADERUNNER / TLOTB : | 330 |
| 4: TUTANKHAMON por YACARE / ZORAN : | 310 |
| 5: FREESTYLE por KRONOS / KRONOS : | 290 |
| 6: ANGEL por DARKNESS / INPHOBIA : | 250 |
| 7: BATMAN por ICEADDER / REQUIEM : | 120 |
| 8: MARUJITA por LEUNAM / NETWORK : | 115 |
| 9: 2 OH CLOCK por HUMPHREY : | 110 |
| 10: CHRISTO por WARLOCK / CAPSULE : | 105 |
| 11: SERPENT por CESAR PORTO/OMEGA : | 105 |

### Graficos 3D

| | |
|---|---|
| 1: CHATEAUX por ASIER HERNAEZ : | 910 |
| 2: CATEDRAL por KUAZAR / GOBLINS : | 715 |
| 3: PARTY por ASIER ISUSI : | 255 |
| 4: TERRAZA por PHASOR / A.CIRCLE : | 210 |
| 5: CASTLE por PITU : | 130 |
| 6: HELLRAISE por BODHY / CLONEZERO : | 90 |
| 7: THE EYE por FLOWRPOWER/THE EYE : | 85 |
| 8: DONDE ESTAN LAS ... por HO-RENDER / THE EYE : | 85 |
| 9: ARAÑA por GUYBRUJA / F.BRAIN : | 80 |
| 10: ERMITA por ANDRES ELIZONDO : | 45 |
| 11: BUZZ por FREDDY : | 40 |
| 12: ROBOTIC por YACARE / ZORAN : | 30 |

Modulos 4K

| | |
|---|---|
| 1: FALSE VISAGE por SMASH / CRYSTAL S. : | 795 |
| 2: NO INSPIRATION por FIREBOY / OZONE : | 550 |
| 3: THE LEGENDS SPREADS por EVELRED / CAPSULE : | 355 |
| 4: SEASINE por CAIN / ARTIFIZIAL : | 280 |
| 5: TOTAL EMPTYNESS por BLUE/NET & HUMPHREY : | 245 |
| 6: FUTURE PARTY por NORK / NETWORK : | 155 |
| 7: VISUAL PATTERNS por JPM / OZONE : | 140 |
| 8: ONE MOMENT PLEASE por CHIP / SOFT.FAILURE : | 130 |
| 9: OUTLINE BLOODNESS por ESTRAYK / CAPSULE : | 125 |
| 10: THE SAY OF ANY LIFE por SERGIO DJ / RL&SDJ : | 105 |
| 11: SPANIARD FEELING por LEUNAM / NETWORK : | 85 |
| 12: I NEED A NEW WORD por HAM / SOFT.FAILURE : | 60 |
| 13: WHEN DREAMS COME... por FALAN / NIVEL 7 : | 25 |

### Multi-Canal

| | |
|---|---|
| 1: STANDARD DEMO SONG por SAMPLEMIND/CLONE 0 : | 295 |
| 2: MOVIMIENTO por NOISYMAN / IGUANA : | 245 |
| 3: KEEPER OF SECRETS por SLAANESH/CRYSTAL S : | 230 |
| 4: INFASHION por CUBICO / DIFTERIA : | 215 |
| 5: WAR 2 por AWESOME / MONSTER : | 210 |
| 6: VALLEY OF CLOUDS por EVELRED / CAPSULE : | 185 |
| 7: BEATA INCONEXA por WALLY / TLOTB : | 185 |
| 8: BLOODY STREETS III por SMASH / CRYSTAL S. : | 170 |
| 9: TRIP por DEM / ZORAN : | 165 |
| 10: L3AD DANC3 por DANIEL DEL REY : | 160 |
| 11: THE SONG OF MY LIFE por SERGIO DJ / RL&SDJ : | 110 |
| 12: SINCRONIZED LIFES por POTXOKI / SYS : | 90 |
| 13: 808 STYLE XM por 808 STYLE : | 50 |
| 14: IMPRESSION por JCL / UNKNOWN : | 35 |

### Intros de 64K (PC)

| | |
|---|---|
| 1: ABDUCTION por TLOTB : | 35 |

### Intros de 4K (PC)

| | |
|---|---|
| 1: REVOLUTION por MARATHON MAN : | 1185 |
| 2: PULSE por ALIEN : | 785 |
| 3: BLAST por UNKNOWN : | 390 |
| 4: KU 3 por BOSKO : | 195 |
| 5: ROTEX por SAIRF : | 105 |

### Fast 2D

| | |
|---|---|
| 1: BICHO por PHORNEE / GOBLINS : | 495 |
| 2: EUSKAL FAST2D 8 por JON MERCERO : | 360 |
| 3: EUSKAL FAST2D 3 por YACARE / ZORAN : | 340 |
| 4: EUSKAL FAST2D 1 por BELCEBU / SYS : | 225 |
| 5: EUSKAL FAST2D 14 por LUIS GARAIKOETXEA : | 215 |
| 6: EUSKAL FAST2D 10 por RIPP / GOBLINS : | 145 |
| 7: EUSKAL FAST2D 2 por T.INFO : | 105 |
| 8: EUSKAL FAST2D 11 por BODHY / CLONEZERO : | 100 |
| 9: EUSKAL FAST2D 9 por JAGO : | 75 |
| 10: TELELOGO por BOSKO / BOSKO : | 50 |
| 11: EUSKAL FAST2D 6 por IMANOL GOMEZ PEREZ : | 45 |
| 12: EUSKAL FAST2D 7 por AITOR O ITURRALDE : | 35 |
| 13: EUSKAL FAST2D 5 por JR : | 30 |
| 14: EUSKAL FAST2D 12 por LORD RAPTOR : | 10 |

### FastMusic

| | |
|---|---|
| 1: PARTY 4 por NOISYMAN / IGUANA : | 605 |
| 2: HUMPAMIX por FIREBOY / OZONE : | 370 |
| 3: EUSKAL FAST NORK por NORK / NETWORK : | 280 |
| 4: FAST 4 por ESTRAYK / CAPSULE : | 265 |
| 5: 1 2 3 REMIX por JCL / UNKNOWN : | 170 |
| 6: THE CHILD SONG por CUBICO / DIFTERIA : | 160 |
| 7: FAST 96 por EVELRED / CAPSULE : | 155 |
| 8: FAST MUSIC BY RAY por JUAN CARLOS JIMENEZ : | 115 |
| 9: FAST 4 por CALI / SERPIX : | 100 |
| 10: POTTOKI HUMPA por POTXOKI / SYS : | 80 |
| 11: SOLO UNA HORA por SLAANESH/CRYSTAL S. : | 65 |
| 12: HP por SHADE / ALIEN : | 60 |
| 13: PIPI por KIWI / SERPIX : | 40 |
| 14: HOT SUKOR por L.A. DE VAZQUEZ : | 25 |

### HumanDemo

| | |
|---|---|
| 1: DIXIE HUMAN DEMO por CAPSULE : | 1085 |
| 2: CUTREX FAST HUMAN por SYS : | 615 |
| 3: NODOR por CRYSTAL SHADE : | 560 |
| 4: HUMAN DOPE 408 por CENTOLOS & MARCIAL : | 555 |
| 5: XPERIMENTAL por FREDDY : | 230 |

### Demos (PC)

| | |
|---|---|
| 1: DESPAIR por IGUANA : | 715 |
| 2: HIPNOTIC por EXOBIT : | 590 |
| 3: BIRTH por INCOGNITA : | 430 |
| 4: ANNUBIS por THE BANNER : | 420 |
| 5: SHEER HEART ATTACK por TLOTB : | 315 |
| 6: 666 por CRYSTAL SHADE : | 240 |
| 7: 18 DAYS por REQUIEM : | 220 |
| 8: SYLOP por CLONE ZERO : | 205 |
| 9: FLASH por UNKNOWN : | 170 |
| 10: VOID por JRQ & TOXIC : | 100 |
| 11: <<DEMO DESCALIFICADA>> : | 95 |
| 12: ASIAN por SAVAGE : | 45 |
| 13: VIRTUAL FLY por DOSIS : | 25 |

**Resultados del concurso.**

# LAS INTERFERENCIAS

*Pedro Antón*

Otro efecto basado en circunferencias. Esta vez, concéntricas.

A lo largo de este curso hemos explotado de diferentes formas la ecuación de la circunferencia. Hemos creado varios tipos de efectos con ellas. Pues bien, en este artículo se verá cómo crear otro asombroso efecto con circunferencias concéntricas, así como algún que otro secreto de la VGA.

## LA VGA

La resolución empleada en este efecto es de 80x50.

El lector recordará cómo al principio de este curso, al implementar el conocido por todos efecto del fuego se comentaba la posibilidad de modificar algunos registros de la VGA, para obtener una resolución menor de 320x200.

Recordemos un poco aquel efecto. Consistía en crear un *buffer* de 160x102 y transformar cada pixel de ese *buffer*. La transformación que estos pixels sufrían era sencillamente engordar. Cada valor del *buffer* era transformado en pixels de 2x2 o bien en pixels de 4x4. Es decir, por cada valor del *buffer* se pintaba un cuadrado de esas dimensiones en la pantalla.

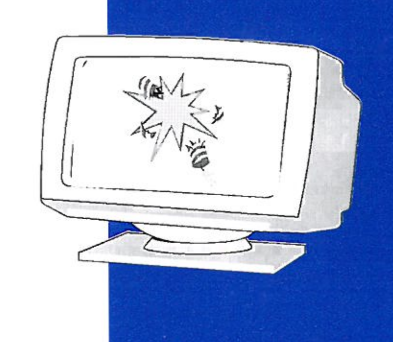
Con las tarjetas de vídeo actuales los famosos modos X han quedado olvidados, los tiempos de acceso a vídeo han mejorado considerablemente, y los modos de vídeo han aumentado tanto en resolución como en definición. Esto no quiere decir que no se deba investigar en la tarjeta de vídeo.

En este artículo se verán unos cuantos registros interesantes de la VGA.

El puerto 3C4h accede, como ya se había visto, al secuenciador de la VGA.

*Índice 2* (lectura/escritura): Máscara de planos.

El bit 0 Activa la escritura al plano 0 si está activo.



**Con las tarjetas de vídeo actuales los tiempos de acceso a vídeo han mejorado considerablemente**

**Este mes se estudia un nuevo efecto para añadir a nuestras demos, el objetivo final es que los lectores que estén siguiendo esta sección desde su primer capítulo, sean capaces de llegar a desarrollar sus propias demos partiendo de los conocimientos en esta sección enseñados y aprovechando al límite su imaginación y originalidad**

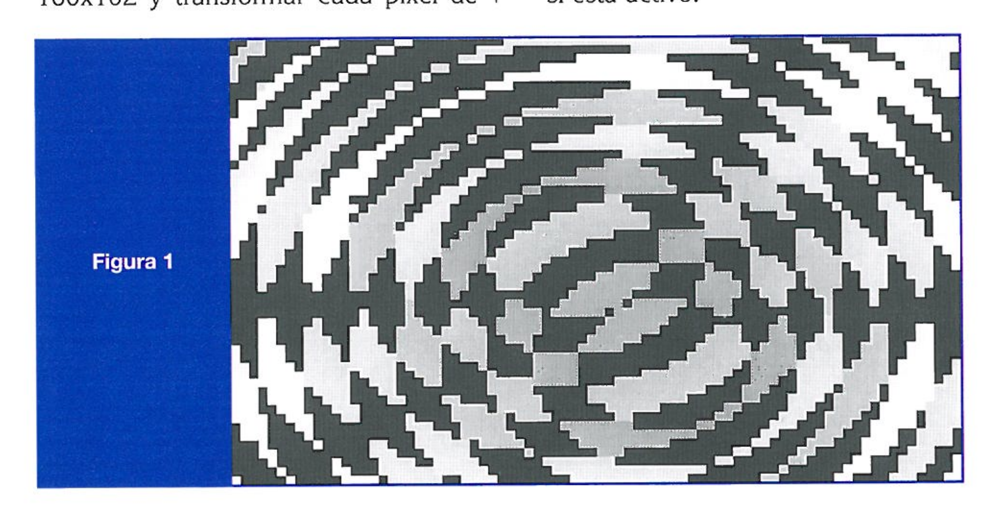


Figura 1

El bit 1 Activa la escritura al plano 1 si está activo.

El bit 2 Activa la escritura al plano 2 si está activo.

El bit 3 Activa la escritura al plano 3 si está activo.

Usado para activar la escritura de los diferentes planos. En el ejemplo que acompaña a este artículo el valor 0Fh activa la escritura en los cuatro planos. De esta forma se accede a cuatro bytes de la memoria de vídeo con una sola escritura. Esto se traducirá a una resolución de 80 pixels de ancho.

*Índice 4* (lectura/escritura): Modo de memoria.

El bit 0 Estará activo en modos de texto.

El bit 1 Activo si la VGA tiene mas de 64Kb.

El bit 2 Si está activo coloca los bytes impares en los planos 1 y 3 y los pares en los planos 0 y 2.

El bit 3 Si está activo los dos últimos bits del *offset* de la memoria de vídeo, controlan el plano donde se va a escribir el dato.

Usado para indicar a la VGA que ahora se va a tomar el control de los planos. En nuestroo ejemplo, toma el valor 06h. Lo mas importante que se ha de destacar de este registro es la desactivación del bit 3.

El puerto 3D4h accede al controlador del tubo de rayos catódicos.

*Índice 9* (lectura/escritura): Máximo número de *scan lines.*

Los bits del 0 al 4 indican el número de *scan lines* por cada barrido horizontal menos uno.

El bit 5 contiene el bit 9 del índice 15 de este mismo puerto.

El bit 6 contiene el bit 9 del índice 18 del puerto 3D4h.

El bit 7 si está activo duplica el scan line.

En el ejemplo adjunto, el valor que toman los 5 primeros bits de este registro es de 7, el resto permanecen inalterados.

*Indice 0Ch* (lectura/escritura): Parte alta de la dirección de comienzo donde el CRTC empieza a mostrar el *buffer* de vídeo.

*Indice 0Dh* (lectura/escritura): Parte baja de la dirección de comienzo donde el CRTC empieza a mostrar el *buffer* de vídeo.

Estos dos registros son usados para el cambio de página activa.

*Indice 14h* (lectura/escritura): Posición del cursor.

Los bits del 0 al 4 indican la posición del cursor dentro de un carácter.

Si el bit 5 está activo cambia la dirección de memoria cada cuatro ciclos.

El bit 6 activa o desactiva el direccionamiento en forma de cuatro bytes.

*Indice 17h* (lectura/escritura): Control de modo.

Si el bit 0 no está activo usa direccionamiento de memoria compatible CGA.

Si el bit 1 no está activo usa direccionamiento de memoria compatible Hercules.

Si el bit 2 está activo incrementa el *scanline* cada dos líneas.

Si el bit 3 está activo incrementa la dirección de memoria cada dos ciclos.

El bit 4 es reservado.

Si el bit 5 está activo y se está en modo *word*, el bit 15 de la dirección

**LISTADO 1**

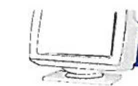
```
Procedure VGA80x50;assembler;
Asm
  mov ah,0Fh
  int 10h
  mov OldMode,al
  mov ax,13h
  int 10h

  cli          { Come oN with Jare's cat & dog mode :}

  mov dx,3C4h     { Port 3C4h                    }
{-------------------------------------------------}
  mov ax,604h     { Index 04-Value 06.           }
  out dx,ax       { Tweak mode init              }
  mov ax,0F02h    { Index 02-Value 0Fh.          }
  out dx,ax       { Enable write to all planes.  }

  mov dx,3D4h     { Port 3D4h                    }
{-------------------------------------------------}
  mov ax,14h      { Index 14h-Value 0            }
  out dx,ax       { Disable dword mode.          }
  mov ax,0E317h   { Index 17h- Value E3h.        }
  out dx,ax       { Enable byte mode.            }
  mov al,9        { Index 09                     }
  out dx,al
  inc dx
  in al,dx        { Read 3D4h register index 09  }
  and al,0E0h     { Set to 0 max. scan lines.    }
  add al,7        { Set 8 scan lines per character row. }
  out dx,al
  sti             { That's all folks! Do you want more }
                  { comments? Write them! X-DDDD  }

  mov ax,0A000h   { ...and now... cLeAr sCrEeN.  }
  mov es,ax
  xor di,di
  db 66h;xor ax,ax
  mov cx,16000
  db 66h;rep stosw
End;
```





**LISTADO 2**

```
PROCEDURE ChangeVGAOff (VGAOff:Word);assembler;
Asm
  mov  dx,3DAh
@@1:            { Wait for vertical retrace start }
  in   al,dx
  test al,8
  jnz  @@1

  mov  bx,VGAOff  { Change video offset...      }
  mov  dx,3D4h    { Port 3D4h                   }
{-----------------------------------------------}
  mov  al,0Ch     { Index 0C                    }
  mov  ah,bh      { Value: High VGAOff byte     }
  out  dx,ax
  inc  ax         { Index 0D                    }
  mov  ah,bl      { Value: Low VGAOff byte      }
  out  dx,ax      { ...video offset changed.    }

  mov  dx,3DAh
@@2:            { Wait for vertical retrace end  }
  in   al,dx
  test al,8
  jz   @@2
End;
```

**LISTADO 3**

```
Procedure MakePalette(Green:Byte);
Var
  CntCol : Byte;
  AuxCol : Byte;
  Up     : Boolean;
Begin
  CntCol :=  1;
  Up     :=True;
  AuxCol :=  24;
  PutColor (0,0,0,0);
  Repeat
    PutColor (CntCol,0,Green,AuxCol);
    If Up Then Begin
           Inc (AuxCol);
           If AuxCol=63 Then Up:=False;
          End
       Else Begin
           Dec (AuxCol);
           If AuxCol=24 Then Up:=True;
          End;
    Inc (CntCol);
  Until CntCol=0;
end;
```

es rotado al bit 0. Si no esta activo y se está en modo *word* el bit 13 es rotado al bit 0.

Si el bit 6 no está activo se está en modo *word*.

Si el bit 7 no está activo el sistema de vídeo permanece inactivo hasta que se active dicho bit.

Para comprender mejor esto es recomendable observar cómo se ha creado la resolución del ejemplo que acompaña a este artículo en el listado 1.

El lector puede intuir ahora la posibilidad de escribir en la memoria de vídeo no visible, ahorrándonos de esta forma el volcado. Pero para ello se debe cambiar el comienzo de la dirección de memoria de la VGA, donde el CRTC empezará a leer. Y como se ha comentado, esto es posible. En el listado 2 se puede observar la forma de hacerlo correctamente.

## EL BUFFER

Se usará parte de la VGA como *buffer*.

El uso de parte de la memoria de vídeo como *buffer* es una de las ventajas de los modos X. De esta forma, además de ahorrar tiempo de volcado, se ahorra la zona de memoria, que normalmente es empleada para el *buffer*.

## LA PALETA

Modificar la paleta aumentará la sensación de movimiento.

El degradado es, como siempre, un buen añadido al efecto. Pero, si a su vez es modificado con el movimiento, el efecto gana vistosidad. El listado 3 muestra cómo se genera la paleta en cada *frame*.

## EL EFECTO

Sumar circunferencias concéntricas es la base de este efecto.

Algo tan sencillo como la suma de distintas partes del *buffer* (similar a lo que se hacía con el plasma) es el alma de este efecto.

La clave está en el *buffer* a diseñar. Se debe diseñar un *buffer* con circunferencias concéntricas con la suficiente resolución. En el ejemplo que acompaña a este artículo la resolución empleada es de

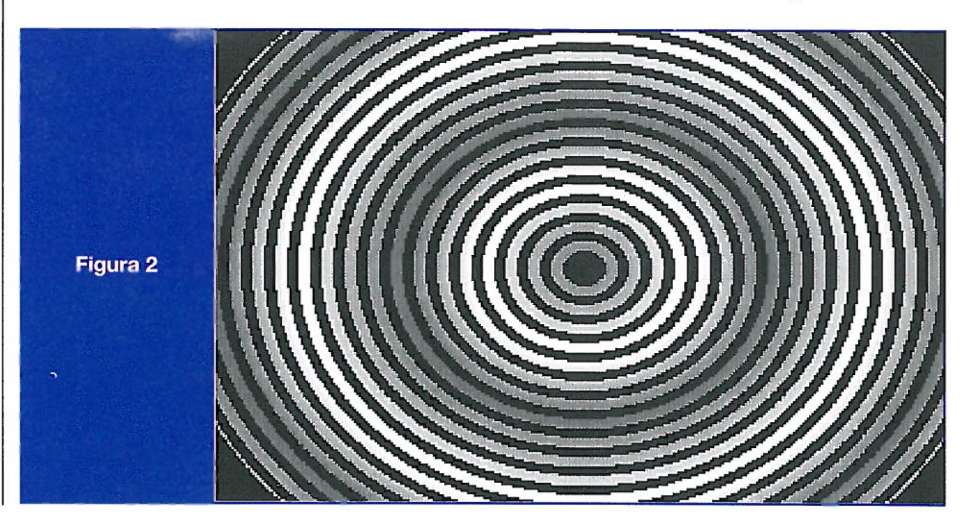


Figura 2

**LISTADO 4**

```
Procedure Interferencias;
 Var
  IniX1,IniY1 : Word;
  IniX2,IniY2 : Word;
  Ang1        : Word;
  Ang2        : Word;
  CntColor    : Byte;
  Up          : Boolean;

 Begin
  OffWrite:=  0;
  OffStart:= 4000;
  Ang1    :=  0;
  Ang2    :=  68;
  CntColor:=  0;
  Up      := True;
  Repeat
   MakePalette  (CntColor);
   If Up then Begin
          Inc (CntColor);
          If CntColor=63 then Up:=False;
         End
       else Begin
          Dec (CntColor);
          If CntColor=1 then Up:=True;
         End;
   IniX1:= 80*FastCos(Ang1);
   asm sar IniX1,8 end;
   IniX1:=160+IniX1;
   IniY1:= 50*FastSin(Ang1);
   asm sar IniY1,8 end;
   IniY1:=100+IniY1;
   Inc (Ang1,4);
   If Ang1>=1024 then Ang1:=Ang1-1024;
   Put80x50Buffer (IniX1,IniY1);

   IniX2:= 80*FastCos(Ang2);
   asm sar IniX2,8 end;
   IniX2:=160+IniX2;
   IniY2:=50*FastSin(Ang2);
   asm sar IniY2,8 end;
   IniY2:=100+IniY2;
   Inc (Ang2,2);
   If Ang2>=1024 then Ang2:=Ang2-1024;
   Put80x50Buffer (IniX2,IniY2);

   If OffWrite=0 then Begin
            OffWrite:=4000;
            OffStart:=0;
           End
         else Begin
            OffWrite:=0;
            OffStart:=4000;
           End;
   ChangeVGAOff  (OffStart);
   FillBytes    ($A000,OffWrite,4000,0);
  Until KeyPressed;
 End;
```

Figura 3

320x200 y el *buffer* es el mostrado en la figura 2.

Como la resolución empleada es de 80x50 será posible moverse con una soltura aceptable por el *buffer* empleado. Pero hay que tener en cuenta un par de detalles para mezclar adecuadamente las partes del *buffer* seleccionadas.

En primer lugar se considerará el color cero como máscara. Es decir, si se lee un byte con valor cero, no se pintará.

En segundo lugar, antes de escribir se comprobará si anteriormente se ha escrito algún valor en el *buffer* distinto de cero, en cuyo caso se escribirá un cero. De esta forma se consigue el bonito entramado que crea este efecto.

Sobre el movimiento que describirá la figura poco hay que decir. Se puede precalcular una secuencia de movimiento, o bien se puede calcular en tiempo real. Esta última opción es la empleada por el ejemplo que acompaña a este artículo, describiendo las partes a mezclar, circunferencias con distinto incremento de ángulo y radio.

El bucle principal, una vez inicializadas las variables, se hará de la siguiente forma:

Se genera la paleta del *frame* actual, dependiendo de la cantidad de verde deseada, dicho valor se almacena en la variable *CntColor*. La variable *Up* se encarga de controlar si dicha variable debe ser incrementada o decrementada. El control de este proceso es realizado una vez se ha generado la paleta.

A continuación se calcula el *offset* del *buffer* que se va a utilizar como primera parte de la mezcla y se escribe en la zona no visible de la VGA.

El siguiente paso es calcular el *offset* del *buffer* que se va a mezclar con el ya escrito y volcarlo a la zona de memoria adecuada, teniendo en cuenta las normas expuestas al principio de este apartado.

Como se observa en el listado 4, entre estos pasos se controlan y actualizan las variables del movimiento.

Para finalizar se modifican las variables *OffWrite* y *OffStart* que controlan el uso de una zona u otra de la VGA usada como *buffer* y se cambia el *offset* de comienzo de la VGA a la zona correcta. Sin olvidar limpiar la zona que se acaba de escribir que en este momento, como era de esperar, no se encuentra visible.

# LOS SERVICIOS DOS

*Enrique de Alarcón*

Si la intención del lector es programar aplicaciones para funcionar bajo DOS que tengan acceso a todos los recursos del sistema, primero tendrá que tener toda la información relacionada con la BIOS y sus servicios (lo cual se dio en los dos números anteriores de la revista, en forma de dos artículos dedicados a la BIOS) y después toda la información posible acerca del propio DOS y sus servicios, la cual se verá en el presente artículo.

Aquellos que usen sólo rutinas (instrucciones) de un compilador para DOS y que crean que no necesitan todo esto, que piensen que prácticamente todos los compiladores tienen algún área del sistema que no queda cubierta con las instrucciones que da el lenguaje en cuestión y que obligatoriamente habrá que usar acceso directo a servicios DOS (como por ejemplo para programar el COM, el ratón u otro periférico). Y de ahí viene que todo buen compilador que se precie debe tener algún sistema para poder acceder a interrupciones, ya sean DOS, BIOS u otras de drivers y demás (por ejemplo, la instrucción *int86x* del Watcom C++).

## EL DOS

La palabra DOS son las siglas del ingles *Disk Operating System* y, a grandes rasgos, es el primer programa que se ejecuta después del arranque de la BIOS (si es que no hay un virus en el sistema) y que está destinado a hacer que el sistema sea operativo para el usuario, encargándose básicamente de permitir el acceso fácil y rápido al disco duro y a los disquetes para poder usar las aplicaciones que se desee. A más bajo nivel se podría decir que es una librería de funciones destinadas a facilitar al programador la creación de aplicaciones.

En el mercado hay varias versiones del DOS de diferentes marcas, más o menos compatibles entre ellas (MS-DOS, DR-DOS, IBM-DOS...). Pero el DOS más extendido en todo el mundo es el MS-DOS de la casa Microsoft, que ha llegado actualmente hasta la versión 6.xx de dicho sistema operativo.

## DATOS GLOBALES

A todos los servicios DOS se accede mediante el uso de interrupciones software que siempre son las mismas, de la misma forma que se hacía con las BIOS, simplemente con otros valores de interrupción. En ensamblador llamar a un servicio sería simplemente usar una instrucción tipo *INT xx*, donde <xx> es el número de interrupción. El número de la función que se quiera ejecutar dentro de dicha INT, si es que posee más de una función (como el caso de la INT 21h), se indicará siempre en el registro AH. Por otra parte, los parámetros que requiera la función se le pasarán usando los registros de CPU que tenga el servicio destinados a ello (AL, DL, BL, BH....BX). Por lo tanto, igual que pasaba con la BIOS, no se necesitará usar la PILA para comunicarse con el DOS.

**DOS es el primer programa que se ejecuta después del arranque de la BIOS**

**En los anteriores números se detallaron todos los secretos y servicios disponibles en la BIOS. En el presente artículo, se hará lo mismo con el DOS, el cual pone disposición del programador una gran librería de funciones que hacen las veces de ampliación de la BIOS**

En lo que a errores se refiere, casi todas las funciones DOS, al volver, retornan la bandera CF=0 si se ha ejecutado correctamente, y CF=1 si se ha producido algún fallo, en cuyo caso también se indicará el código de error en AX. En la Tabla 1 se detalla una lista con los códigos de error que usa el DOS en sus funciones y su correspondiente significado. Como se puede observar, casi todos los errores están destinados a informar sobre accesos a disco o sobre control de la red (Network). En total hay 100 errores distintos posibles y se detallan en la tabla 1.

En el DOS, todos los servicios básicos estándar se encuentran englobados dentro de la interrupción 21h (excepto unos pocos que también se mencionarán), a diferencia de la BIOS, donde cada área del sistema, se cubría usando una interrupción diferente (por ejemplo recordar que todo lo relacionado con pantalla estaba en la INT 10h, y lo referente a disco se encontraba en la INT 13h).

## ÁREAS DE FUNCIONES

Como ya se ha comentado, todas las funciones disponibles del DOS están contenidas dentro de la interrupción 21h, y están agrupadas secuencialmente según su utilidad.

A continuación se especifican los tipos de funciones que contiene:

1.- Entrada/Salida de carácteres por pantalla/teclado.
2.- Control de directorio.
3.- Control de disco.
4.- Control de ficheros.
5.- Control de ficheros (método FCB).
6.- Control asignación de memoria.
7.- Funciones del sistema.
8.- Control de proceso.
9.- Funciones de almacenamiento.
10.- Funciones de almacenamiento por método FCB.
11.- Control de hora y fecha.
12.- Máquina/Impresora.
13.- Redireccionamiento de Dispositivos.

## DESCRIPCIÓN DE INTERRUPCIONES DOS

A continuación se describirán todos los servicios agrupados por clases de utilidad, detallando su utilidad, parámetros de entrada, de salida, posibles errores y datos de interés que puedan tener asociadas.

**1.- Funciones orientadas I/O de carácteres:**

- *AH=01h*. Leer carácter+*Echo*: Lee una pulsación del teclado escribiendo el carácter leído en pantalla. Si no hay tecla disponible, espera. Entrada: No hay. Salida: AL=Carácter ASCII leído.
- *AH=02h*. Escribe carácter: Escribe el carácter indicado en la posición actual del cursor en pantalla. Entrada: DL=Carácter a escribir. Salida: No hay.
- *AH=03h*. Leer carácter de un COM: Lee el carácter actual pendiente en el *buffer* del puerto serie, que por defecto es el COM1. Entrada: No hay. Salida: AL=Carácter leído.
- *AH=04h*. Escribe carácter en COM: Escribe el carácter indicado en el puerto serie, que por defecto es el COM1. Entrada: DL=Carácter. Salida: No hay.
- *AH=05h*. Escribe *char* en Dispositivo de salida: Escribe el carácter indicado en el puerto serie, que por defecto es el LPT1. Entrada: DL=Carácter. Salida: No hay.
- *AH=06h*. I/O de *char* en dispositivo estándar: Escribe o lee desde el dispositivo estándar de I/O un carácter. Esta rutina evita cualquier posible interrupción por parte del DOS en la operación. Entrada: DL=FFh para leer, cualquier otro valor para escribir. Salida: AL=Carácter si era lectura.
- *AH=07h*. Leer carácter sin *Echo*: Lee un carácter del dispositivo de entrada estándar (normalmente teclado) sin filtrar y sin escribirlo por pantalla. Si no hay tecla disponible, espera. Entrada: No hay. Salida: AL=Carácter ASCII leído.
- *AH=08h*. Leer carácter sin *Echo*: Lee un carácter del dispositivo de entrada estándar (normalmente teclado) filtrándolo y sin escribirlo por pantalla. Si no hay tecla disponible, espera. Entrada: No hay. Salida: AL=Carácter ASCII leído.
- *AH=09h*. Escribe cadena: Escribe la cadena ASCII indicada en pantalla. Entrada: DS:DX=Segmento/*offset* al inicio de la cadena que se desee escribir. Salida: No hay.
- *AH=0Ah*. Lee cadena del teclado: Lee una cadena de bytes del *buffer* del teclado y lo almacena en la dirección indicada. Entrada: DS:DX=Segmento:*offset*. Salida: No hay.
- *AH=0Bh*. Chequeo de disponibilidad: Chequea si hay un carácter disponible en el *buffer* del teclado. Entrada: No hay. Salida: AL=00h si caracter no disponible. Otro valor si hay caracteres disponibles.
- *AH=0Ch*. Limpia *buffer* de teclado: Limpia el buffer del teclado de pulsaciones pendientes y llama a la función pendiente. Entrada: AL=Función a ejecutar después. Salida: La correspondiente a la función indicada por el usuario (ver anteriores).

**2.- Control de directorio:**

- *AH=39h*. Crea directorio: Crea un directorio en la unidad y PATH indicados. Entrada: DS:DX=Segmento:*offset* con la cadena ASCII que contiene el *PATHname* del directorio que se quiera crear. Salida: CF=1 si error -> AX=Código de error.
- *AH=3Ah*. Borrar directorio: Borra el directorio especificado. Entrada: DS:DX=Puntero a cadena ASCII con pathname del directorio a borrar. Salida: CF=1 si error ->AX=Código de error.
- *AH=3Bh*. Cambia directorio: Cambia el directorio activo por el especificado. Entrada: DS:DX=Puntero a cadena ASCII con *pathname* del directorio a situar. Salida: CF=1 si error ->AX=Código de error.
- *AH=47h*. Extraer *pathname*: Obtiene el *pathname* del directorio donde nos encontremos actualmente. Entrada: DL=Unidad de disco, DS:SI=*buffer* de 64 bytes. Salida: CF=1 si error ->AX=Código de error.

**3.- Funciones de disco:**

- *AH=0Dh*. Reset de *buffers*: Limpia todos los datos pendientes de los *buf-*

**En el DOS, todos los servicios básicos estándar se encuentran englobados dentro de la interrupción 21h**

*fers* de disco escribiéndolos físicamente al disco, no dejando así ninguno pendiente. Entrada: No hay. Salida: No hay.

• *AH=0Eh*. Cambia unidad activa: Cambia la unidad de disco activa de trabajo. Entrada: DL=Unidad de disco (0=A, 1=B...). Salida: AL=Número de unidades del sistema.

• *AH=19h*. Obtener unidad de disco activa: Devuelve cuál es la unidad de disco que está activa al ser llamada. Entrada: No hay. Salida: AL=Unidad de disco activa.

• *AH=1Bh/1Ch*. Lee información del disco: Obtiene información de la unidad de disco actual/especificada. Entrada: DL=Unidad si AH=1Ch. Salida: AL=Sectores por pista, DS:BX=Tabla de datos físicos, CX=Sectores físicos, DX=Pistas. Si error AL=FFh.

• *AH=2Eh*. Cambia *Flag* Verificación: Pone a 1 o a 0 la bandera para verificaciones post-escrituras de disco que realiza el DOS. Entrada: AL=0 si no se quiere verificar, 1=si se desearealizar la verificación. Salida: No hay.

• *AH=36h*. Lee información de la unidad indicada: Lee información útil de la unidad de disco quese le indique. Entrada: DL=Unidad de disco. Salida: AX=Sectores por pista, BX=Número de pistas libres, CX=Bytes por sector, DX=Pistas por disco.

• *AH=54h*. Obtiene *Flag* Verificación: Devuelve el estado activo o inactivo del *Flag* interno del DOS para comprobación de escrituras en disco. Entrada: No hay. Salida: 00=Inactiva, 1=Activa.

**4.- Operaciones sobre ficheros:**

• *AH=3Ch*. Crear fichero: Crea el fichero en el *Pathname* indicado. Entrada: CX=Bits atributos de apertura (*Read-only*, reservado...), DS:DX=Segmento:*Offset* con el *Pathname*. Salida: AX=*Handle* al fichero.

• *AH=3Dh*. Abre fichero: Abre el fichero especificado por el *pathname* indicado. Entrada: AL=Modo de acceso. Salida: AX=*Handle* al fichero.

• *AH=3Eh*. Cierra fichero: Cierra el fichero especificado. Entrada: BX=*Handle* del fichero. Salida: No hay.

• *AH=41h*. Borra fichero: Borra el fichero especificado. Entrada: DS:DX=Puntero al *Pathname* con el fichero que se quiere borrar. Salida: No hay.

• *AH=43h*. Obtiene/modifica atributos: Obtiene o modifica los atributos actuales del fichero indicado. Entrada: AL=0 si obtener, 1=cambiar, CX=Nuevos Atributos (si AL=1), DS:DX=Puntero al *pathname* del fichero. Salida: CX= Atributos.

• *AH=45h*. Duplica *Handle*: Crea un segundo puntero de fichero identico al indicado. Entrada: BX=Puntero al fichero duplicado. Salida: AX=Nuevo *handle*.

• *AH=46h*. Redireccionar *Handle*: Cambia el puntero hacia el cual apunta el segundo *handle* indicado al mismo fichero/posición que el primero. Entrada: BX=*Handle* original, CX=*Handle* a redireccionar. Salida: No hay.

• *AH=4Eh*. Encuentra primer fichero: Encuentra el primer fichero disponible en el directorio actual de disco. Entrada: CX=Atributos a usar para buscar, DS:DX=Segmento:*Offset* con el puntero al path donde buscar. Salida: Se almacenan los datos en la DTA del sistema operativo.

• *AH=4Fh*. Busca siguiente fichero: Busca el fichero siguiente al primero encontrado con la anterior función dentro del directorio. Entrada: No hay. Salida: Se almacena en el DTA.

• *AH=56h*. Renombrar fichero: Cambia el nombre del fichero indicado por otro. Entrada: DS:DX=*Pathname* del fichero a modificar, ES:DI=*Pathname* nuevo para el fichero. Salida: No hay.

• *AH=57h*. Obtener/Cambia Hora-Fecha fichero: Obtiene o cambia la hora y fecha del fichero indicado. Entrada: BX=*Handle* al fichero, si AL=1 CX=Nueva hora, DX=Nueva fecha. Salida: Si AL era 0, CX=Hora, DX=Fecha.

• *AH=5Ah*. Crea fichero temporal: Crea un fichero temporal con los atributos indicados. Entrada: CX=Atributos, DS:DX=Puntero al nombre del fichero a crear. Salida: AX=*Handle*, DS:DX=Segmento:Offset con el *pathname* completo al fichero.

• *AH=5Bh*. Crea nuevo fichero: Crea un nuevo fichero sólo si no existe ya uno con el mismo nombre y path. Entrada: CX=Atributos del fichero a crear, DS:DX=Puntero al *Pathname* del fichero a crear. Salida: AX=*Handle*.

• *AH=67h*. Cambia número de *Handle*: Cambia el número máximo de ficheros que puede haber abiertos simultáneamente en el sistema. Entrada: BX=Número de ficheros abiertos posibles. Salida: No hay.

**5.- Operaciones con ficheros usando FBC:**

El método FBC de acceso a ficheros es el sistema que proporcionó el DOS desde que apareció hasta la versión 2.00 para acceder a los ficheros de disco y que se basaba en que cada fichero abierto posee en memoria un FCB o bloque de control de fichero donde está toda la información referente a él (en vez de un *Handle*). A la hora de realizar cualquier operación sobre el fichero, siempre se deberá proporcionar a la función la dirección donde se encuentra el FCB. Los bloques FCB no los da el DOS al abrir un fichero, sino que deben ser creadospor el programador.

Un dato muy importante de este sistema es que, si finaliza la ejecución de la aplicación quedando ficheros abiertos con este sistema, el DOS no se encargará de cerrar los ficheros pendientes, y si hubiese datos pendientes de escritura se perderían.

Debido a esto último y a que es bastante pesado manejar los ficheros con éste método, se dejó en desuso a partir de la versión 2.00, donde apareció el concepto de *Handles*.

A continuación de detallan las funciones destinadas a este sistema:

• *AH=0Fh*. Abre fichero: Abre el fichero indicado en el FCB seleccionado. Entrada: DS:DX=Puntero al inicio del FCB indicado. Salida: No hay.

• *AH=10h*. Cierra fichero: Cierra el fichero especificado en el FCB indicado. Entrada: DS:DX=Puntero al inicio del FCB indicado. Salida: No hay.

• *AH=11/12h*. Busca fichero: Busca el primer/siguiente fichero dentro del directorio indicado: Entrada: DS:DX=Segmento:*offset* con puntero a FCB donde buscar. Salida: FCB contiene fichero encontrado.

• *AH=13h*. Borra fichero: Borra el fichero indicado. Entrada: DS:DX=Puntero al FCB con el fichero que queremos borrar. Salida: No hay.

• *AH=14h*. Crea fichero: Entrada: DS:DX=Puntero al inicio del FCB con los datos del fichero a crear. Salida: No hay.

• *AH=17h*. Renombra fichero: Cambia el nombre del fichero indicado.

Entrada: DS:DX=Puntero al inicio del FCB con el/los fichero(s) a modificar. Salida: No hay

• *AH=23h*. Obtener longitud de fichero: Almacena la longitud del fichero indicado dentro del FCB. Entrada: DS:DX=Puntero al inicio del FCB donde está el fichero del que se quiere encontrar la longitud. Salida: FCB modificado.

**6.- Gestión de memoria:**

• *AH=48h*. Crear bloque: Crea un bloque nuevo de memoria libre con el tamaño indicado. Entrada: BX=Número de parágrafos del bloque requerido (bloques de 16 bytes). Salida: AX=Segmento al inicio del bloque creado.

• *AH=49h*. Libera bloque: Libera el bloque de memoria indicado. Entrada: ES=Segmento del bloque a liberar. Salida: No hay.

• *AH=4Ah*. Modifica bloque: Modifica el tamaño actual del bloque de memoria especificado. Entrada: BX=Nuevo tamaño del bloque, ES=Segmento del bloque a modificar. Salida: No hay.

• *AH=58h*. Obtiene/Modifica método de almacenaje: Permite obtener o cambiar el método que usa el DOS a la hora de buscar nuevos espacios de memoria para asignar bloques de memoria. Hay 3 métodos: 00h=Primera localización, 01h=Mejor localización, 02h=Última localización. Entrada: AL=0, obtener método; si AL=1, BX=Método a usar (0h-2h). Salida: AX=Método si AL era 0.

**7.- Funciones del sistema:**

• *AH=25h*. Modifica Interrupción: Cambia un vector de interrupción para que apunte donde se le indique. Entrada: AL=Interrupción a modificar, DS:DX=Nuevo vector. Salida: No hay.

• *AH=30h*. Obtener versión del DOS: Entrada: No hay. Salida: AL=Versión mayor, AH=Versión menor, BH=OEM's manufacturador, BL:CX=Número serial.

• *AH=33h*. Obtener/Modificar *Flag BREAK*: Obtiene/Modifica el *Flag* que usa el DOS para saber si debe comprobar o no la pulsación de *breaks* mediante CTRL+C durante operaciones. Entrada: AL=0 obtener, AL=1 cambiar, DL=00 si desactivar, DL=01h si activar. Salida: Si AL era 0, DL=Estado actual.

• *AH=34h*. Dirección InDOS: Devuelve un puntero tipo FAR que apunta a la bandera InDOS del sistema, la cual indica cuándo se están ejecutando funciones del sistema operativo (1=Función DOS ejecutándose). Entrada: No hay. Salida: ES:BX=Puntero a InDOS.

• *AH=35h*. Obtiene vector INT: Obtiene el puntero que contiene el vector de interrupción indicado. Entrada: AL=Número de interrupción. Salida: ES:BX=Segmento:*Offset* que contiene el vector.

• *AH=38h*. Obtiene/Cambia *Country*: Obtiene o cambia la información internacional del país almacenada en el sistema. Entrada: Si AL=0 obtener información del propio país, DS:DX=Puntero a *buffer* donde almecanar información. Salida: Contenido del *buffer*.

• *AH=44h*. Funciones Control I/O: Conjunto de subfunciones para control de procesos de entrada y salida. Entrada: AL=Subfunción, que va desde 00h hasta 0Fh. Cada subfunción posee parámetros propios. Salida: Depende de la subfunción.

• *AH=50h*. Cambia dirección PSP: Cambia el segmento que apunta al *buffer* de 256 bytes con el PSP del programa. Entrada: BX=Nuevo segmento para el PSP. Salida: No hay.

• *AH=51h*. Obtiene dirección PSP: Obtiene el segmento que apunta al *buffer* de 256 bytes con el PSP del programa. Entrada: No hay. Salida: BX=Segmento que apunta al PSP.

• *AH=59h*. Obtener información extra: Obtiene más información precisa del último error que se ha producido en la ejecución de una función DOS. Entrada: BX=00h. Salida: AX=Código de error extendido, BH=Clase de error, BL=Acción recomendable, CH=Tipo de error, ES:DI=Volúmen del disco a insertar.

• *AH=5Eh*. Funciones de Impresora: Conjunto de 3 subfunciones para gestión de impresora. Sus opciones son: Obtener nombre máquina, cambiar cadena setup-impresora y obtener cadena setup-impresora. Entrada: AL=Número de subfunción, 00h-02h. Otros parámetros dependen de la subfunción. Salida: Depende de la subfunción.

• *AH=5Fh/AL=02h*. Obtiene lista de Redirecciones: Devuelve una lista con los dispositivos del sistema lógicos presentes y sus *network files*/directorios o impresoras hacia donde están apuntando. Entrada: AL=02h, BX=Índice de la lista de redirección, DS:SI=Puntero a *buffer* 16 bytes donde almacenar nombre de dispositivo, ES:DI=Puntero a *buffer* de 128 bytes donde almacenar el *network name* a donde apunta. Salida: BH=Estado dispositivo, BL=Tipo dispositivo, CX=Parámetros activos, DX=Destruido, BP=Destruido, DS:SI/ES:DI=No alterados.

• *AH=5Fh/AL=03h*. Redirecciona dispositivo: Entrada: BL=Tipo de dispotivo, CX=Parámetros de llamada, DS:SI=Puntero al nombre ASCIIZ local, ES:DI=Puntero al *network name* ASCIIZ (incluye password). Salida: No hay.

• *AH=63h*. Obtener Ptr Tabla extendida ASCII: Obtiene el puntero a la tabla que contiene la lista de bytes permitidos para usar como caracteres extendidos. Esta función no se detalla más por no estar presente a partir de la versión 3.0 del DOS.

**8.- Funciones de control de proceso:**

• *AH=00h*. Termina ejecución: Finaliza la ejecución de la aplicación en curso. Entrada: CS=Segmento que apunta al PSP de la aplicación. Salida: No hay.

• *AH=26h*. Crear nuevo PSP: Copia el PSP del programa actual al *buffer* de memoria que se le indique. Entrada: DX=Segmento donde está el *buffer* donde copiar el PSP. Salida: No hay.

• *AH=31h*. Terminar + Residente: Finaliza la ejecución de la aplicación actual, devolviendo un código de retorno y dejando parte de la aplicación bloqueada en memoria. Entrada: AL=Código de retorno, DX=Memoria a reservar (bloquear) para la aplicación a dejar residente (en parágrafos). Salida: No hay.

• *AH=4Bh*. Ejecuta programa: Ejecuta una programa tipo EXE sin salir de la propia aplicación para volver tras finalizar. Entrada: AL=Subfunción (00h carga y ejecuta / 03h carga Overlay), ES:BX=Puntero a bloque de parámetros, DS:DX=Puntero a especificaciones de programa. Salida: Todos los registros se destruyen excepto CS:IP.

• *AH=4Ch*. Fin ejecución + Código retorno: Finaliza la aplicación igual que la función 00h, devolviendo además un código de retorno. Entrada: AL=Código a devolver. Salida: No hay.

• *AH=4Dh*. Obtiene Código retorno: Tras ejecutar un programa, se usará esta función para obtener el código de retorno que se haya devuelto. Entrada: No hay. Salida: AH=Código.

• *AH=62h.* Obtiene el segmento del PSP: Obtiene el segmento que apunta al inicio del PSP de la aplicación que se está ejecutando. Entrada: No hay. Salida: BX=Segmento del PSP.

**9.- Funciones de almacenamiento:**

• *AH=1Ah.* Sitúa dirección del DTA: Indica al sistema donde está el área de transferencia de disco que usarán las funciones de FCB del DOS. Entrada: DS:DX=Puntero donde está el DTA. Salida: No hay.

• *AH=2Fh.* Obtiene dirección del DTA: Obtiene el puntero al área de transferencia de disco que usan las funciones de FCB del DOS. Entrada: No hay. Salida: ES:BX=Puntero al DTA.

• *AH=3Fh.* Lee de dispositivo: Lee información de la posición actual del dispositivo al cual apunta el *handle* que se le indique. Entrada: BX=Handle, CX=Número de bytes a leer, DS:DX=Puntero destino donde almacenar la información leída. Salida: No hay.

• *AH=40h.* Escribe en dispositivo: Escribe información en la posición actual del dispositivo al cual apunta el *handle* que indicado. Entrada: BX=Handle, CX=Número de bytes a escribir, DS:DX=Puntero destino donde está almacenada la información. Salida: No hay.

• *AH=42h.* Cambia puntero de I/O: Cambia el *offset* del puntero de escritura/lectura del *handle* que se le indique. Entrada: AL=Método (00h *offset* absotuto, 01h desde posición actual, 02h desde final de fichero), BX=Handle a modificar, CX=Parte alta del nuevo *offset*, DX=parte baja del nuevo offset. Salida: No hay.

• *AH=5Ch.* Bloquea/Desbloquea región: Permite bloquear o desbloquear una zona determinada de un fichero abierto. Entrada: AL=00h si bloquear, 01h si desbloquear, BX=Handle, CX=Parte alta del *offset*, DX=Parte baja del Offset, SI=Parte alta de la longitud de la región a alterar, DI=Parte baja de la longitud. Salida: No hay.

• *AH=68h.* Finaliza *handle*: Se escriben el contenido de todos los *buffers* pendientes en disco y se actualizan fecha y hora del fichero correspondiente al *handle* indicado. Entrada: BX=*Handle*. Salida: No hay.

**10.- Funciones de almacenamiento FCB**

• *AH=14h.* Lectura secuencial: Lee el siguiente bloque secuencial del fichero e incrementa el puntero de I/O. Entrada: DS:DX=Segmento:*Offset* del FCB abierto de donde leer. Salida: AL=00 si lectura correcta, 01h si fin de fichero, 02h si segmento finalizado, 03h si lectura parcialmente en fin de fichero.

• *AH=15h.* Escritura secuencial: Escribe en el siguiente bloque secuencial del fichero e incrementa el puntero de I/O. Entrada: DS:DX=Puntero al FCB del fichero abierto donde escribir. Salida: AL=Resultado escritura, 00h si correcto, 01h si disco lleno, 02h si segmento finalizado.

• *AH=21h.* Lectura aleatoria: Lee el bloque seleccionado del fichero abierto indicado en memoria. Entrada: DS:DX=Puntero al FCB del fichero abierto. Salida: AL=Resultado lectura (igual que en la función 14h).

• *AH=22h.* Escritura aleatoria: Escribe en el bloque seleccionado del fichero abierto indicado. Entrada: DS:DX=Puntero al FCB del fichero abierto. Salida: AL=Resultado lectura (igual que en la función 14h).

• *AH=24h.* Cambia RRN: Cambia el Bloque de lectura/escritura actual del FCB indicado que corresponde a la posición actual de I/O del fichero. Entrada: DS:DX=Puntero al FCB a modificar. Salida: AL=Destruido.

• *AH=27h.* Lectura aleatoria múltiple: Lee uno o mas bloques de datos en memoria de la posición deseada de un fichero abierto. Entrada: CX=Número de bloques a leer, DS:DX=Puntero al FCB del fichero del que leer. Salida: AL=Resultado lectura, CX=Bloques leídos.

• *AH=28h.* Escritura aleatoria múltiple: Escribe uno o más bloques de datos en la posición deseada de un fichero abierto. Entrada: CX=Número de bloques a escribir, DS:DX=Puntero al FCB del fichero en el que escribir. Salida: AL=Resultado escritura, CX=Bloques escritos.

**11.- Funciones control Fecha y hora:**

• *AH=2Ah.* Obtener fecha: Devuelve la fecha actual almacenada en el sistema. Entrada: No hay. Salida: CX=Año (1980-2099), DH=Mes del año, DL=Día, AL=Día de la semana (0=Domingo).

• *AH=2Bh.* Cambiar fecha: Permite modificar la fecha almacenada en el sistema. Entrada: CX=Nuevo año, DH=Mes, DL=Día. Salida: AL=00h si fecha correcta, Ffh si fecha no válida.

• *AH=2Ch.* Otener hora: Devuelve la hora actual almacenada en la BIOS. Entrada: No hay. Salida: CH=Hora, CL=Minutos, DH=Segundos, DL=Centésimas de segundo.

• *AH=2Dh.* Cambiar hora: Permite cambiar la hora actual almacenada en la BIOS. Entrada: CH=Hora, CL=Minutos, DH=Segundos, DL=Centésimas de segundo. Salida: (Error igual que en la función 2Bh).

## OTROS SERVICIOS DOS

Como ya se ha dicho, todo el núcleo de servicios del DOS se encuentra incluido en la interrupción 21h, pero hay además algunos servicios secundarios sueltos que se encuentran en otros números de interrupción y que son los siguientes:

• *INT 20h.* Terminar programa: Finaliza la ejecución del programa actual. Si el retorno se hace hacia el DOS, se efectúan las siguientes operaciones: Primero se libera toda la memoria que ocupaba, acto seguido se resetean todos los *buffers* de disco que hubiese y se cierran todos los ficheros que quedasen abiertos y por último se restauran los vectores de Interrupción 22h, 23h y 24h. Entrada: CS=Segmento al PSP del programa a finalizar. Salida: No hay.

• *INT 22h.* Puntero de finalización: Este vector de INT apunta al inicio de la rutina que se ejecutará al finalizar la ejecución de la aplicación actual.

• *INT 23h.* Puntero CTRL+C: Apunta a la rutina que se ejecuta cuando se fuerza un *break* por teclado mediante CTRL + C.

• *INT 24h.* Puntero Error Crítico: Este vector apunta a la rutina que ejecuta el DOS cuando se produce un error crítico del sistema (generalmente un error hardware). El ejemplo más visto es cuando se intenta acceder a un disquete y éste no se encuentra en la unidad.

• *INT 25h.* Lectura absoluta de disco: Lee del disco usando acceso directo a

**TABLA DE ERRORES PARA LAS FUNCIONES**

01h - Número de función no válido.
02h - Fichero no encontrado.
03h - Path (ruta de acceso) no encontrado.
04h - Demasiados ficheros abiertos.
05h - Acceso denegado.
06h - Puntero (Handle) inválido.
07h - Control de bloques de memoria destruido.
08h - Memoria insuficiente.
09h - Dirección de memoria inválida.
0ah - Entorno inválido.
0bh - Formato inválido.
0ch - Código de acceso inválido.
0dh - Dato inválido.
0eh - Unidad desconocida.
0fh - Unidad de disco inválida.
10h - No se pudo borrar directorio.
11h - No es el mismo dispositivo.
12h - No hay mas ficheros.
13h - Disco protegido contra escritura.
14h - Unidad desconocida.
15h - Disco no preparado.
16h - Comando desconocido.
17h - Error de datos (en CRC).
18h - Longitud de estructura errónea.
19h - Seek Error.
1ah - Medio desconocido.
1bh - Sector no encontrado.
1ch - Impresora sin papel.
1dh - Escritura imposible.
1eh - Lectura imposible.
1fh - Error general.
20h - Violación de compartición.
21h - Violación de bloqueo.
22h - Cambio de disco inválido.
23h - FCB no disponible.
24h - Búffer de compartición excedido.
25h - Código de página error.
26h - Handle llegó a fin de fichero.
27h - Disco lleno.
28h/31h - Reservados.
32h - Petición Network no soportada.
33h - Máquina remota no preparada.
34h - Nombre de Network duplicado.
35h - Nombre de Network no encontrado.
36h - Network ocupada (busy).
37h - Dispositivo ya no existe en Network.
38h - Comando de NetBIOS ha excedido límite.
39h - Error hardware del Network.
3ah - Respuesta incorrecta desde la Network.
3bh - Error en Network.



3ch - Adaptador remoto incompatible.
3dh - Impresión excedida.
3eh - Insuficiente espacio para imprimir el fichero.
3fh - El fichero a imprimir fue borrado.

40h - Nombre borrado de la Network.
41h - Acceso denegado a Network.
42h - Tipo de dispositivo erróneo.
43h - Nombre de network no encontrado.
44h - Limite de nombre del Network excedido.
45h - Sesión NetBIOS excedida.
46h - Pausa temporal.
47h - Petición no atendida en Network.
48h - Impresión o acceso a disco redireccionado.
49h/4Fh - Reservados.
50h - El fichero ya existe.
51h - FCB duplicado.
52h - No se puede crear el directorio.
53h - Fallo en la Int 24h.
54h - Demasiados directorios.
55h - Redirección duplicada.
56h - Código de acceso incorrecto.
57h - Parámetros inválidos.
58h - Fallo en escritura dentro de la Network.
59h - Función no soportada en la Network.
5ah - Componente necesario no disponible.
65h - Dispositivo no seleccionado.

sectores físicos mediante BIOS. Entrada: AL=Unidad, CX=Número de sectores a leer, DX=Sector inicial donde leer, DS:BX=Búffer donde almacenar la información. Salida: No hay.

• *INT 26h*. Escritua absoluta de disco: Escribe en el disco usando acceso directo a sectores mediante BIOS. Entrada: AL=Unidad, CX=Número de sectores a escribir, DX=Sector inicial donde escribir, DS:BX=*Buffer* donde está almacenada la información. Salida: No hay.

• *INT 27h*. Fin + Residente: Finaliza la ejecución de la aplicación actual y deja parte residente. Este método de residente se usa para crear el sistema de carga por overlays. Entrada: CS=Segmento del bloque a dejar residente, DX=Longitud en bytes + 1 del segmento a dejar. Salida: No hay.

• *INT 28h*. Fin + Residente: Deja residente la aplicación actual, pero con la posibilidad de que se reactive en cualquier momento si se produce alguna circunstancia especial. Este método de residente lo usan programas como los antivirus residentes o diccionarios permanentes, que se activan desde el DOS con la pulsación de teclas especiales. Entrada: No hay. Salida: No hay.

• *INT 2Fh*. Interrupción Multiplexada: Permite comunicarse con programas residentes como APPEND, SHARE y ASSIGN. Debido a la complejidad del tema, sólo se referencian, sin entrar en detalles.

***Subfunciones disponibles:***

• *AH=00h*. Print Spooler.

• *AH=06h*. Comunicación con ASSIGN. Entrada: AL=00. Salida: AL=Estado de ASSIGN (0=No instalado pero posible, 1=No instalado ni posible, Ffh instalado).

• *AH=10h*. Comunicación con SHARE. Entrada: AL=00h. Salida: AL=Estado de SHARE (Igual que ASSIGN).

• *AH=B7h*. Comunicación con APPEND. Entrada: AL=Subfunción (04h=Coger versión de APPEND, 06h=Obtener función, 07h=Seleccionar función, 11h=Cambia El nombre de retorno). Salida: Igual que los anteriores.

## BIBLIOGRAFÍA

*8088-8086/8087 Programación ENSAMBLADOR en entorno MS-DOS, Miguel Ángel Rodríguez Roselló. Anaya Multimedia.

*Funciones del MS-DOS, Anaya Multimedia/Microsoft Press.

*Introducción al MS-DOS, Grupo Waite; O'Day, Kate.

## EN RESÚMEN

En el presente artículo se han detallado todos los servicios que posee el DOS dentro de la interrupciónes 21h a 28h, incluyendo los de acceso a disco, pantalla, puertos, etc... Toda esta información, junto a la aparecida ya en los dos artículos dedicados a la BIOS, dan una completa referencia técnica acerca de los recursos que proporciona el sistema sobre el que se programa, eliminando cualquier limitación de acceso y programación que se pudiera tener al hardware estándar instalado.

# VARIABLES, CONSTANTES Y MATRICES

*Juan Manuel y Luis Martín*

La forma en que un lenguaje de programación almacena datos es a través de variables y constantes. En el primer caso, el dato puede ser alterado a lo largo de la ejecución del programa, mientras que en el segundo el dato permanece inalterado. Como ya se mencionó en el artículo anterior, antes de utilizar variables en una aplicación es necesario informar al ordenador del número y el tipo de éstas. A esta operación se le denomina declaración, y en Visual Basic se reliza mediante la sentencia *Dim*.

Aunque Visual Basic ofrece la posibilidad de ir añadiendo variables a medida que van siendo necesarias, sin declararlas inicialmente, es conveniente informarlas al principio del programa o procedimiento en que son utilizadas. Si en la sección de declaraciones del módulo se incluye la sentencia *Option Explicit*, Visual Basic generará un mensaje de aviso cada vez que encuentre una variable no declarada. Esta sentencia puede ser incluida automáticamente en los nuevos módulos que se añadan al proyecto. Para ello, debe seleccionarse la opción *Opciones* del menú *Herramientas* y activar la casilla *Declaración de variables* requerida en la ficha *Entorno*, tal y como se muestra en la figura 1.

La declaración de una variable también supone su inicialización a un determinado valor. Este valor depende del tipo de datos con el que se ha declarado la variable, tal y como se muestra en la tabla 1.

## ÁMBITO Y VISIBILIDAD

Las variables se clasifican, además de por el tipo de datos que contienen, por otras características que determinan su comportamiento durante la ejecución del programa. Estas características indican dónde y cuándo va a existir una variable, así como los lugares del código desde los que son accesibles.

Todas las variables que intervienen en una aplicación se crean reservando el espacio que van a ocupar en memoria durante la ejecución de la aplicación. Con algunas variables esta operación se realiza al comenzar la ejecución, mientras que con otras se realiza durante la ejecución. Además, la destrucción de algunas variables se realiza también durante la ejecución de la aplicación, de forma que el espacio ocupa-

**En el artículo anterior se analizaron las variables y constantes de *Visual Basic*, atendiendo al tipo de datos que éstas pueden contener. En LA presente entrega se analizan también estos conceptos, aunque atendiendo a dos nuevos aspectos como son el ámbito y la visibilidad. Además, Visual Basic dispone de un tipo especial de variables, denominadas matrices, que permiten almacenar gran cantidad de información de forma ordenada.**

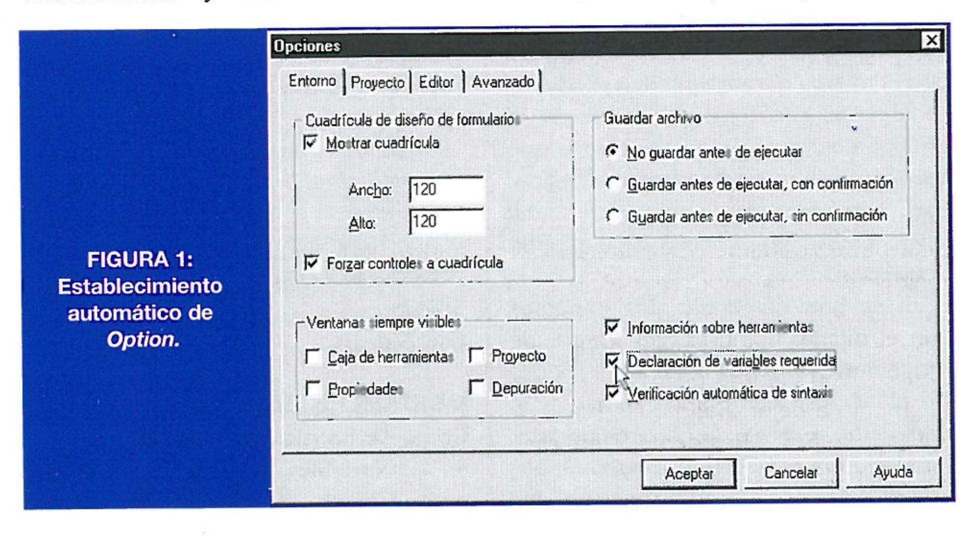



**FIGURA 1: Establecimiento automático de *Option.***

| TABLA 1 | |
|---|---|
| Tipo de datos | Valor Inicial |
| Numéricos | 0 |
| *Boolean* | *False* |
| *Strings* de longitud variable | Cadena vacía ("") |
| *Strings* de longitud fija | Espacios en blanco |
| *Variant* | *Empty* |
| Tipos definidos por el usuario | Se inicializa cada subtipo |

**Valores de iniciales de las variables.**

do por ellas en memoria queda liberado. Sin embargo, existen otras variables que se destruyen cuando la ejecución de la aplicación termina.

El tiempo de vida de una variable, es decir, el intervalo de tiempo transcurrido desde su creación hasta su destrucción, determina lo que se conoce como ámbito de una variable. En Visual Basic existen dos posibles ámbitos para las variables, en función del lugar donde éstas hayan sido declaradas:

- ***Nivel de Procedimiento***: Es el ámbito para las variables declaradas dentro de los procedimientos. Estas variables son creadas automáticamente al comenzar el procedimiento y su destrucción se realiza también de forma automática cuando éste finaliza.
- ***Nivel de Módulo***: Es el ámbito de las variables declaradas en la sección de declaraciones de un módulo. Las variables se crean de forma automática cuando comienza la ejecución de la aplicación y se destruyen al finalizar la misma.

Por su parte, la zona de la aplicación donde se puede acceder a una variable para conocer o modificar su contenido determina lo que se conoce como visibilidad de la variable. Mediante esta característica es posible la existencia de variables con un mismo nombre, aunque en zonas de visibilidad distintas, sin que en un principio se produzcan errores por ello. En Visual Basic existen dos tipos de variables en función de su visibilidad:

- ***Privadas***: Son aquellas variables a las que sólo es posible acceder desde el procedimiento o módulo en que han sido declaradas.
- ***Públicas***: Son aquellas variables a las que se puede acceder desde cualquier parte de la aplicación.

Teniendo en cuenta los dos tipos de ámbito y los dos de visibilidad, podrían obtenerse cuatro tipos de variables. Sin embargo, las variables declaradas a nivel de procedimiento son siempre privadas, pudiendo ser accedidas únicamente desde el procedimiento en que han sido declaradas.

A nivel de procedimiento, las variables son siempre privadas y se declaran con la sentencia *Dim*. Cuando se utiliza esta sentencia a nivel de módulo, las variables serán por defecto de tipo privada. Sin embargo, es posible utilizar las sentencias *Private* y *Public* para indicar explícitamente el tipo de visibilidad. Su sintaxis es la misma que la de la sentencia *Dim*:

*Private Variable [As Tipo]*
*Public Variable [As Tipo]*

## OCULTACIÓN Y CUALIFICACIÓN

Cuando se utilizan variables declaradas con el mismo nombre pueden producirse algunos conflictos derivados de la ambigüedad surgida en relación a qué variable es la que está siendo accedida. Para resolver estos conflictos, Visual Basic realiza un proceso denominado ocultación.

Si se han declarado dos variables con el mismo nombre, una a nivel de procedimiento y otra a nivel de módulo, el acceso siempre se refiere a la declarada a nivel de procedimiento, quedando la otra oculta momentáneamente. Del mismo modo, si existen dos variables declaradas a nivel de módulo, una privada y otra pública de otro módulo, el acceso siempre se refiere a la privada, quedando la otra oculta momentáneamente.

También pueden producirse conflictos cuando en los módulos de formularios se declaran variables con el mismo nombre que una propiedad o un control, aunque esto no es posible con variables declaradas a nivel de módulo. En este caso, la variable tiene preferencia de acceso.

Para poder acceder entonces a la propiedad debe utilizarse la cualificación de la misma. Para ello, hay que especificar el nombre de formulario (o la palabra reservada *Me*) delante del nombre de la propiedad, separando ambos por el operador "!". En el siguiente ejemplo puede verse este tipo de comportamiento:

```
Sub Form_Load ()
  Dim Left 'Existe una propiedad con ese nombre
  Left = 0 'Acceso a la variable
  Form1!Left = 0 'Acceso a la propiedad
  Me!Left = 0 'Acceso a la propiedad
End Sub
```

**Visual Basic realiza un proceso denominado ocultación para resolver estos conflictos**

También puede producirse un conflicto cuando se accede a la propiedad predeterminada de un control y existe una variable declarada con el mismo nombre que el control. En este caso, la ambigüedad puede resolverse cualificando el control con el nombre del formulario.

```
Sub Form_Load()
  Dim Label1
  Label1 = "" 'Acceso a la variable
  Form1!Label1 = "" 'Acceso a la propiedad
  Label1.Left = 0 'Provoca un error
  Me!Label1.Left = 0 'Acceso a la propiedad
End Sub
```

## VARIABLES ESTÁTICAS

Como se ha mencionado anteriormente, las variables declaradas a nivel de procedimiento se crean al comienzo de

**CUADRO 1**

1. Añadir al formulario por defecto dos botones de comando y una etiqueta, situándolos de forma similar a como se muestra en la figura 2.

2. Especificar en tiempo de diseño las siguientes propiedades:

| | | |
|---|---|---|
| *Form1* | *Name* | *frmEstática* |
| | *Caption* | *"Variable estática"* |
| *Command2* | *Name* | *cmdSalir* |
| | *Caption* | *&Salir* |
| *Command3* | *Name* | *cmdClic* |
| | *Caption* | *Clic* |
| | *Font.Bold* | *True* |
| *Label1* | *Name* | *lblClics* |
| | *Alignment* | *2 'Center* |
| | *Caption* | *Número de clics realizados: 0* |

3. Escribir los siguientes procedimientos de evento:

```
Private Sub cmdClic_Click()
  Static NumClics As Integer
  NumClics = NumClics + 1
  lblClics.Caption = "Número de clics realizados:" & NumClics
End Sub

Private Sub cmdSalir_Click()
  Unload Me
End Sub
```

4. Salvar el formulario en el archivo *ESTATICA.FRM* y el proyecto en *ESTATICA.VBP*.

**Ejemplo de variable estática.**

la ejecución del procedimiento y se destruyen al finalizar ésta, perdiéndose los valores almacenados en ellas. Este comportamiento puede alterarse utilizando la sentencia *Static* en la declaración, en lugar de *Dim*, de la siguiente forma:

*Static Variable [As Tipo]*

El cuadro 1 contiene un sencillo ejemplo que ilustra este tipo de comportamiento. Si en dicho ejemplo la variable se declara con la sentencia *Dim*, el valor visualizado en todas las llamadas será 1, ya que siempre será inicializada.

Por su parte, las variables declaradas a nivel de módulo siempre tienen un comportamiento de tipo estático, ya que se crean al comienzo de la ejecución de la aplicación y sus valores son preservados durante toda ella. En el caso de los formularios, el valor es preservado incluso cuando el formulario ha sido descargado de la memoria.

## CONSTANTES

En determinadas ocasiones es necesario utilizar repetidamente un valor literal que, por su gran extensión o su complicación para recordar, puede resultar incómodo de manejar. En este caso puede utilizarse una contante en vez del valor literal. Se trata de una representación de dicho valor mediante un nombre simbólico, que facilita su manipulación a lo largo del código. Aunque el aspecto y manejo de las constantes es similar al de las variables, no es posible asignar un valor a una constante salvo en la definición de la misma. Una vez asignado este valor, permanecerá inalterado a lo largo de toda la ejecución.

**En Visual Basic existen dos posibles ámbitos para las variables**

Visual Basic dispone de dos tipos distintos de constantes que pueden ser utilizadas a lo largo el código. Por un lado están las constantes intrínsecas, que se encuentran definidas en las librerías del sistema, y por otro las constantes simbólicas, que deben ser declaradas por el programador.

Las constantes intrínsecas son nombres simbólicos que se encuentran definidos en las librerías de objetos de las aplicaciones y en los controles. Las librerías *Visual Basic* (*VB*), *Visual Basic para Aplicaciones* (*VBA*) y *Objetos de Acceso a Datos* (*DAO*) proporcionan una serie de constantes que pueden ser utilizadas a lo largo del código.

Otras aplicaciones externas, como Microsoft Excel o Microsoft Word, suministran también sus propias constantes. Todas ellas pueden utilizarse en el código y consultarse mediante el examinador de objetos. Las constantes de cada librería suelen utilizar un prefijo identificativo. Así, por ejemplo, las de las librerías *VB* y *VBA* comienzan con el prefijo *vb*, las de la librería *DAO* con *db* y las de Microsoft Excel con *xl*.

*vbOkCancel*
*dbOpenDynaset*
*xlDialogBorder*

Los controles personalizados también pueden incluir sus propias constantes. Así, por ejemplo, las constantes del control Multimedia (*MCI*) comienzan con el prefijo *mci*, y las del control cuadrícula con *grd*.

*mciModePlay*
*grdAlignLeft*

Las constantes simbólicas son nombres declarados en el código que representan un determinado valor. En Visual Basic, la declaración de este tipo de constantes sigue las mismas reglas que la de variables, utilizando para ello la sentencia *Const*:

*[Public|Private] Const NombreConstante [As Tipo] = Expresión*

El valor asignado al nombre se especifica mediante una expresión que puede contener valores literales, otras constantes previamente definidas o combinaciones de ambos mediante operadores aritméticos y lógicos. No es posible utilizar nombres de variable ni llamadas a funciones. Además, es posi-

**CUADRO 2**

1. Añadir al formulario un cuadro de texto y dos botones de comando, situándolos de forma similar a como se muestra en la figura 3.

2. Especificar en tiempo de diseño las siguientes propiedades:

```
Form      Name       frmMatriz
CaptionMatriz dinámica
Text1     Name       txtElem
Text
Command1             Name       cmdAñadir
Caption&Añadir nuevo
Command2             Name       cmdExtraer
Caption&Extraer último
Enabled              False
```

3. Escribir las siguientes declaraciones y procedimientos de evento:

```
Dim Matriz() As String, NElem As Integer

Private Sub cmdAñadir_Click()
  NElem = NElem + 1
  ReDim Preserve Matriz(NElem)
  Matriz(NElem) = txtElem.Text
  txtElem.Text = ""
  cdmExtraer.Enabled = True
  txtElem.SetFocus
End Sub

Private Sub cmdExtraer_Click()
  txtElem.Text = Matriz(NElem)
  NElem = NElem - 1
  ReDim Preserve Matriz(NElem)
  If NElem = 0 Then cdmExtraer.Enabled = False
  txtElem.SetFocus
End Sub
```

4. Salvar el formulario en el archivo *MATRIZ.FRM* y el proyecto en *MATRIZ.VBP*.

**Ejemplo de matrices dinámicas.**

tipo de datos que va a contener. La única excepción es que la matriz esté contenida en una variable de tipo *Variant*, en cuyo caso puede hacerse uso de la cláusula *As* en la sentencia *ReDim*.

*Dim Num() As Integer, Matriz As Variant*
*...*
*ReDim Num(1 To 10)*
*...*
*Matriz = Num*
*ReDim Matriz(1 To 10) As Double*

Cada vez que se realiza un proceso de redimensionamiento mediante la sentencia *ReDim*, los valores almacenados hasta ese momento en la matriz se pierden. Para conservar los valores almacenados en la matriz puede hacerse uso de la cláusula *Preserve*. Sin embargo, la utilización de esta cláusula sólo permite cambiar la cota superior de la última dimensión. En caso contrario se producirá un error en tiempo de ejecución.

*ReDim Num(10, 15, 20)*
*ReDim Preserve Num(10, 15, 25)*
*'Operación satisfactoria*
*ReDim Preserve Num(25, 15, 25)*
*'Provoca un error*

Para facilitar el trabajo con matrices dinámicas, Visual Basic dispone de dos funciones que permiten conocer las cotas inferior y superior de cualquier dimensión de una matriz. Su sintaxis es la siguiente:

*UBound( NombreMatriz [, Dimensión])*
*LBound( NombreMatriz [, Dimensión])*

La dimensión cuyo tamaño se desea consultar debe ser especificada como un número entero que indica el orden de la misma en la matriz. Si se omite este argumento, el valor por defecto es 1.

*ReDim Matriz(1 To 9)*
*'Para aumentar el tamaño por arriba y por abajo (de 0 a 10)*
*ReDim Matriz(LBound(Matriz) - 1 , UBound(Matriz) + 1)*

La utilización de matrices dinámicas optimiza el manejo de la memoria, ya que permite crear matrices con un número de elementos ajustado a las necesidades de cada momento, siendo posible liberar el espacio ocupado por aquellos elementos que no están siendo utilizados.

El tamaño máximo de las matrices en Visual Basic solamente está limitado por la capacidad de memoria disponible en el sistema. En sistemas de 32 bits, como Windows'95 y Windows NT, los valores máximo y mínimo de las cotas se encuentran entre -2^31 y 2^31.

El cuadro 2 contiene un sencillo ejemplo ilustrativo del manejo de matrices dinámicas. Se trata de una aplicación que permite añadir y extraer elementos de una matriz dinámica de cadenas a partir de un cuadro de texto.

## CONCLUSIÓN

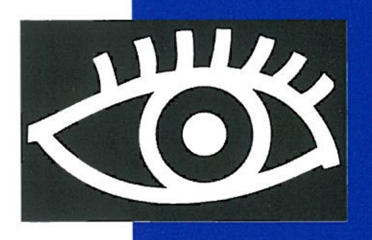
La correcta comprensión de ámbito y visibilidad, tanto para variables y constantes como para matrices, permitirá evitar muchos errores en la codificación de las aplicaciones. Para ello es conveniente que el programador realice diversos ejemplos hasta obtener la soltura deseada en su manejo.

En el próximo artículo se analizarán las distintas estructuras de que dispone Visual Basic para controlar el flujo del programa, así como el correcto manejo de los distintos tipos de procedimientos.




# EL CONTROL DE WINDOWS 95 *LIST VIEW*

*Jorge R. Regidor*



En el artículo anterior se hizo un repaso global a uno de los nuevos controles introducidos en Windows 95, el control *Tree View*. En esta entrega se va a realizar un repaso similar a otro de los nuevos y más potentes controles estándar de Windows 95, el control *List View*.

Los programas creados para Windows 95, así como el propio sistema, utilizan frecuentemente este control. Uno de los más claros ejemplos es la propia *shell* de Windows 95 (Ver figura A), que usa el control *List View* para mostrar y colocar los diversos iconos y carpetas de programas. Otro ejemplo de una aplicación que hace uso de este control es Microsoft Exchange (Ver figura B), donde se muestran todos los mensajes dentro de un control *List View*.

## ¿QUÉ SON LOS CONTROLES *LIST VIEW*?

El control *List View* es una lista de elementos, los cuales se componen de una etiqueta, una imagen, el estado del elemento, valores definidos por la aplicación y diversos subelementos.

Las imágenes de los diferentes elementos pueden ser mostrados de varias formas diferentes. Existen cuatro formas de mostrar la lista de elementos con imágenes asociadas. La primera muestra los elementos con imágenes grandes, la segunda muestra los elementos con imágenes pequeñas, la tercera muestra los elementos en una lista vertical también con imágenes pequeñas y la cuarta y última muestra un informe de los elementos con imágenes pequeñas en una columna vertical, añadiendo además la información de los subelementos asociados.

Como se ha adelantado anteriormente, cada elemento puede tener información complementaria asociada en forma de subelementos, la cual puede visualizarse sólo en la vista de informe en forma de columnas.

Cada control *List View* a su vez contiene otros controles como son el con-

**Una de las grandes mejoras introducidas en Windows 95 es el explorador del sistema, que sustituye al administrador de archivos. Este programa basa toda su potencia en la utilización del control *List View* para visualizar los diferentes ficheros y directorios que existen en los diversos discos.**

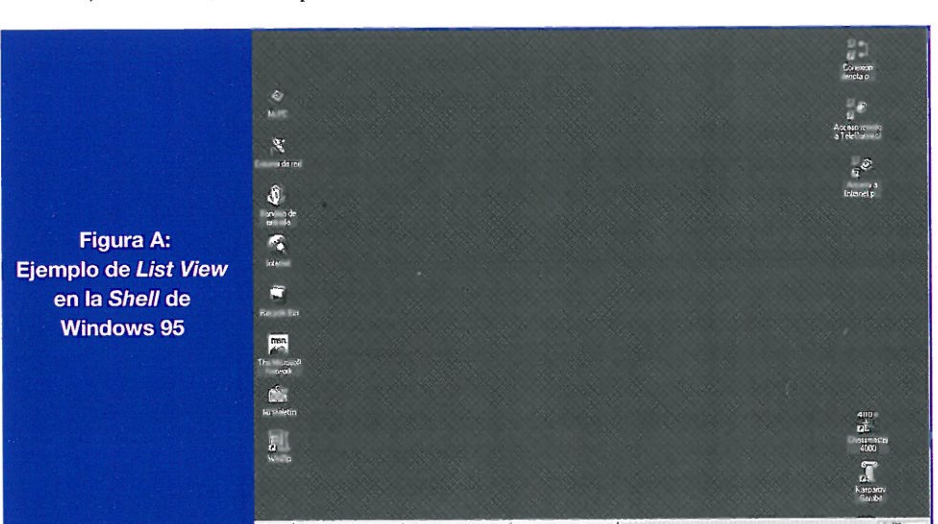


**Figura A: Ejemplo de *List View* en la *Shell* de Windows 95**

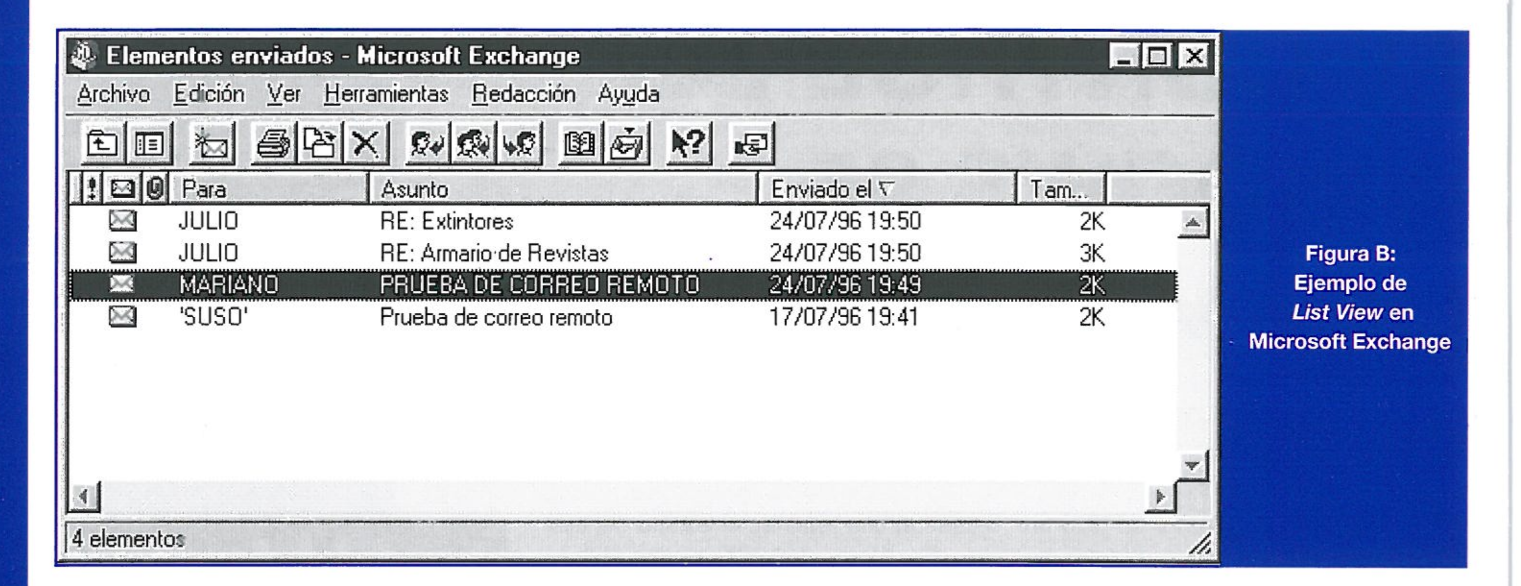



**Figura B: Ejemplo de *List View* en Microsoft Exchange**

trol *Image List* y el control *Header*. El control *Image List* contiene la lista de todas las imágenes que podrán contener cada uno de los elementos. Puede haber dos o tres listas de imágenes asociadas a un control *List View*, cada una de ellas con una labor específica. Una contiene las imágenes para los elementos cuando se muestran los iconos grandes (normalmente 32x32 pixels), otra contiene las imágenes para mostrar los iconos pequeños (normalmente 16x16) y la tercera contiene una lista de imágenes para asignarla a cada icono en función del estado actual de cada elemento.

## LA CLASE *CLISTCTRL*

Como todo el mundo supone, en la librería de clases MFC existe una clase que encapsula la funcionalidad del control *List View*. Esta clase se denomina *CListCtrl*.

Esta clase suministra una serie de funciones miembro para acceder de forma más sencilla a los atributos y operar sobre los controles *List View*, así como para crearlos y mostrarlos. Además, y fruto de la integración con *ClassWizard*, permite automatizar notablemente la atención a determinados eventos que sobre el control sucedan.

La clase *CListCtrl* debe ser utilizada siempre que se quiera crear un control que sea hijo de una ventana principal o de una caja de diálogo. Si se desea que este control sea la vista de la aplicación, la forma más sencilla es indicarlo desde *AppWizard*, lo que se traduce en crear una clase del tipo *CListView*, mucho más adecuada para este tipo de aplicaciones.

Para esta clase existe un sólo constructor y sin parámetros, lo cual lo hace muy sencillo de utilizar. Pero, como ya se sabe, el hecho de crear un objeto *ClistCtrl* no implica que el control *List View* sea creado. Es decir, con el constructor de *CListCtrl* se crea el objeto que controla el acceso a los servicios del control, pero no el control (la ventana en pantalla) en sí. Para ello se debe llamar a la función miembro *Create*, que se ocupa de esta última labor. En el programa ejemplo posterior todo esto no es necesario, ya que al haber incluido el control dentro de una caja de diálogo, es Windows el encargado de crear el control cuando crea el diálogo.

La función *Create* tiene el siguiente formato:

- *BOOL Create( DWORD dwStyle, const RECT& rect,*
- *CWnd* pParentWnd, UNIT nId);*

donde *dwStyle* define el estilo del control que se va a crear. Los diferentes estilos se pueden ver posteriormente, *rect* es una referencia a una estructura del tipo *RECT* que contiene las coordenadas que definen el tamaño del control, *pParentWnd* es un *handle* que apunta a la ventana padre que crea el control y *nId* es el identificador del control.

## ESTILOS DEL CONTROL *LIST VIEW*

Además de todos los estilos comunes a todas las ventanas, el control *List View* tiene unos estilos específicos que permiten definir varios comportamientos totalmente diferentes. Estos estilos son:

- *LVS_ALIGNLEFT*: Especifica que los iconos estarán alineados a la izquierda en la vista de iconos grandes y pequeños.

**Los programas creados para Windows 95, así como el propio sistema, utilizan frecuentemente el control *List View***

- *LVS_ALIGNTOP*: Especifica que los iconos estarán alineados a la izquierda en las vista de iconos grandes y pequeños.
- *LVS_AUTOARRANGE*: Especifica que los iconos se organizarán automáticamente en las vista de iconos grandes y pequeños.
- *LVS_BUTTON*: Implica que los iconos serán vistos como botones en la vista de iconos grandes.
- *LVS_EDITLABELS*: Permite que las etiquetas de los elementos puedan ser editadas.
- *LVS_ICON*: Activa la vista de iconos grandes.

• *LVS_LIST*: Activa la vista de lista de iconos en columnas.
• *LVS_NOCOLUMNHEADER*: Indica que el control *header* que se muestra por defecto en la vista de informe no será mostrado.
• *LVS_NOITEMDATA*: Reserva sólo memoria para almacenar el estado de cada elemento. Los demás campos como la etiqueta, la imagen, los textos de los subelementos o los datos de aplicación. Esto implica que el programa debe atender el mensaje de notificación *LVN_GETDISPINFO* para suministrar a cada elemento la información que requiera.
• *LVS_NOLABELWRAP*: Muestra la etiqueta de cada elemento en una sola línea. Por defecto se utilizan varias líneas.
• *LVS_NOSCROLL*: Deshabilita el uso las barras de *scroll*. Esto implica que los elementos deberían estar siempre visibles en el área de cliente, ya que sino no se podrá acceder a ellos.
• *LVS_NOSORTHEADER*: Este estilo determina que los cabeceras del control *header* no sean botones, es decir, no generen eventos de click. Es útil si estos elementos no van realizan ninguna acción.
• *LVS_OWNERDRAWFIXED*: Implica que la ventana dueña del control recibirá mensajes *WM_DRAWITEM* para dibujar cada uno de los elementos del control.
• *LVS_REPORT*: Activa la vista de informe.
• *LVS_SHAREIMAGELISTS*: Indica que el control *List View* no es el propietario de los controles *Image List* asociados a él. Esto es útil cuando se quieren compartir las listas de imágenes entre varios controles *List View*.
• *LVS_SINGLESEL*: No permite la múltiple selección de elementos.
• *LVS_SMALLICON*: Activa la vista de iconos pequeños.
• *LVS_SORTASCENDING*: Ordena los elementos de forma ascendente basándose en la etiqueta de los elementos.
• *LVS_SORTDESCENDING*: Ordena los elementos de forma descendente basándose en la etiqueta de los elementos.

Si se activan algunos de los dos últimos estilos, no se debe suministrar la función *callback* para ordenar elementos. Este método para ordenar elementos se verá más adelante.

**Mediante AppWizard se pueden crear aplicaciones con vistas basadas en el control ListView**

## LA CLASE *CLISTVIEW*

Antes se ha comentado la posibilidad de utilizar la clase *CListView* frente a la clase *CListCtrl* cuando lo que se desea es que la aplicación contenga un control *List View* como vista principal. Para ello *AppWizard* permite automatizar todo el proceso seleccionando el tipo de vista en la última su ventana (Ver figura C).

La clase *CListView* deriva de la clase *CView*, la cual encapsula todas las operaciones que sobre una vista se pueden realizar. Además, esta clase contiene a la clase *ClistCtrl*, la cual es accesible desde la única función miembro propia de *CListView*, que es *GetListCtrl*. Esta función devuelve una referencia al control asociado con la vista. Con dicha referencia se puede acceder a todas las funciones miembro de la clase *CListCtrl*.

**Cada control *List View* contiene a su vez otros controles como son el control *Image List* y el control *Header***

## USO DE *CLISTCTRL* DESDE CAJAS DE DIÁLOGO

La forma más sencilla de utilizar cualquier control es dentro de una caja de diálogo, y el control *List View* no va a ser menos. Para añadir un control *List View* basta con seleccionar el icono del control de la barra de controles (Ver figura D). El aspecto de éste es como si fuera una pequeña ventana. Al seleccionarlo, el cursor cambia de forma para que se indique el punto donde se va a situar. Una vez hecho esto, aparece el control como se ve en la figura D. Para poder editar sus propiedades y estilos a aplicar, basta con hacer un doble click sobre él y aparecerá una ventana como la que se muestra de nuevo en la figura D.

Una vez que se ha insertado el control en la caja de diálogo hay que realizar el enlace con el código en C++, que permitirá modificar los atributos de

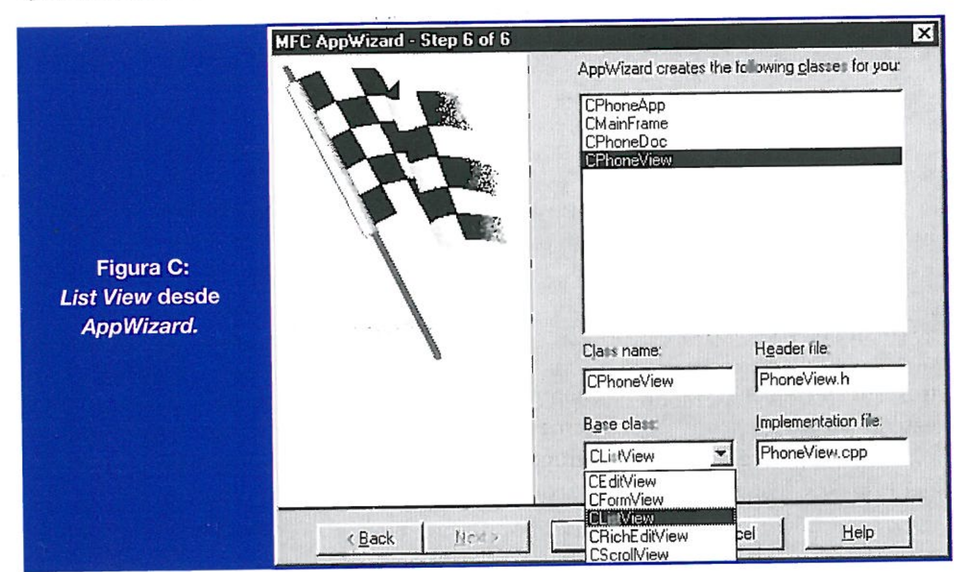



**Figura C:** *List View* desde *AppWizard*.

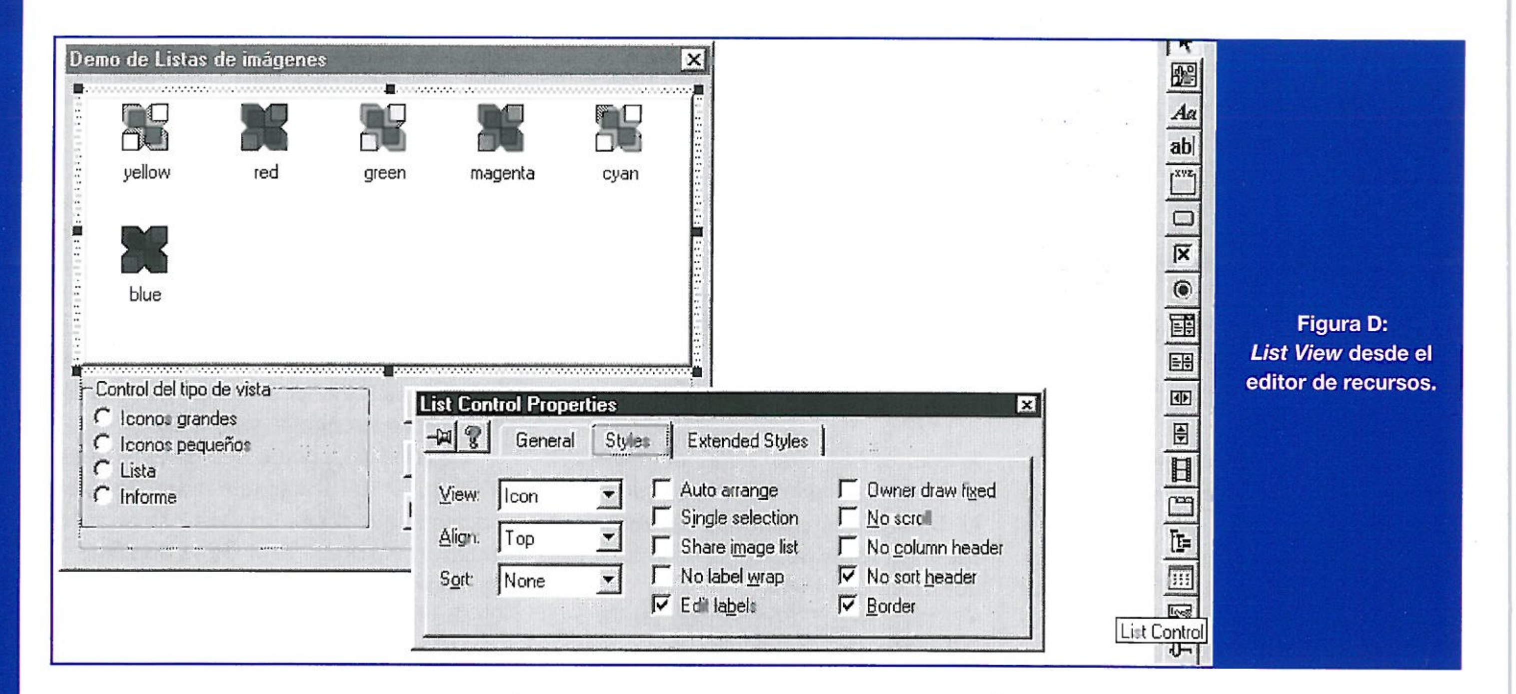



**Figura D:** ***List View* desde el editor de recursos.**

control, operar sobre él y atender sus mensajes de notificación. Para ello se selecciona el control y se pulsa el botón de *ClassWizard*. Como resultado de esto se muestra la ventana de *ClassWizard*, que va a permitir automatizar todo el proceso de programación del control *List View*. En esta ventana hay dos secciones que interesan principalmente. La primera es *Message Maps*, que permitirá automatizar el proceso para atender a los diferentes mensajes de notificación del control, y la segunda sección es *Member Variables*, que va a permitir automatizar la definición de una variable *CListCtrl*, perteneciente a la clase que encapsula el diálogo, y su enlace con el control *List View* creado desde el editor de recursos. Este método es alternativo a las llamadas a la función miembro *GetDlgItem*, permitiendo además la transferencia automática de los datos del diálogo al crearse y destruirse.

**La forma más sencilla de utilizar cualquier control es dentro de una caja de diálogo, y el control List View no va a ser menos**

Vamos a continuar con la sección *Member Variables*, ya que la sección *Message Maps* ya ha sido tratada en otros artículos. Al entrar en esta sección se observa una lista con todos los identificadores de controles que están contenidos en el diálogo (Ver figura E), y haciendo doble click sobre *IDC_LIST_COLOR* (nuestro control *List View*) se muestra otra ventana que permite indicar el nombre que se quiere para la variable, la categoria a la que pertenece, en este caso un control, y por último el tipo de variable a crear, que en este caso es *CListCtrl*. Si se indica el nombre de la variable y se pulsa Aceptar, se creará una variable miembro de la clase *CListDemoDlg* que permitirá acceso total al control. El proceso que permite todo esto va a ser obviado para ser expuesto en siguientes entregas, ya que no es objetivo de este artículo profundizar en ese tema.

## EL PROGRAMA DE EJEMPLO

Para ilustrar el funcionamiento de la clase *CListCtrl* se ha creado una aplicación llamada *ListDemo*, la cual tiene una vista en diálogo y que permite realizar diversas operaciones sobre un control *List View*, con el objetivo último de exponer los métodos de programación de este control y su clase asociada. En el programa se muestra cómo asociar las listas de imágenes, cómo añadir elementos, editar sus etiquetas, ordenar los elementos, saber el número de elementos y poner las diferentes vistas al control.

## LISTAS DE IMÁGENES DEL CONTROL *LIST VIEW*

Como se dijo con anterioridad, un control *List View* puede tener asociadas tres listas de imágenes. Una para mostrar los iconos grandes, otro para mostrar los iconos pequeños y otra para mostrar el estado de cada elemento.

Existe una función miembro que permite asociar una lista de imágenes con el control. El formato de esta función es el siguiente:

*CImageList* SetImageList( CImageList* pImageList, int nImageList );*

donde *pImageList* es un puntero a la lista de imágenes que se quiere asociar y *nImageList* indica a donde se va a asociar la lista. Los posibles valores de este campo son:

- *LVSIL_NORMAL*: Lista de imágenes para mostrar los iconos grandes.
- *LVSIL_SMALL*: Lista de imágenes para mostrar los iconos pequeños.
- *LVSIL_STATE:* Lista de imágenes para mostrar las imágenes de estado.

En el programa ejemplo existen dos listas de imágenes creadas a partir de unos iconos hechos desde el editor de recursos. Una lista de imágenes tiene

un tamaño de 32x32 pixels y la otra de 16x16 pixels, cada una de ellas con 9 imágenes que se corresponden con un círculo, un cuadrado y un rectángulo en los colores rojo, azul y verde.

Después de insertar cada elemento se establece la etiqueta de los subelementos, que en el caso de nuestro programa es uno y permite hacer una descripción del elemento. Además, y para permitir luego la ordenación se llena el campo de datos con el número del elemento, llamando a la función *SetItemData*.

**Cada elemento puede tener varios subelementos asociados, los cuales se muestran sólo en la vista de informe**

## ELEMENTOS, SUBELEMENTOS Y COLUMNAS

Como ya se ha comentado, cada elemento puede tener varios subelementos asociados, los cuales se muestran sólo en la vista de informe. Cada subelemento consta de una cadena de texto, la cual se muestra en cada columna del control *header* asociado. Para crear cada columna existe una función miembro cuyo formato es:

*int InsertColumn( int nCol, LPCTSTR lpszColumnHeading,*
*int nFormat = LVCFMT_LEFT, int nWidth = -1,*
*int nSubItem = -1 );*

donde *nCol* es el índice de la columna, *lpszColumnHeading* es el título de la columna, *nWith* es el ancho de la columna, *nSubItem* es el elemento asociado con esta columna y *nFormat* indica el alineamiento de la columna. Los posibles valores son:

- *LVCFMT_LEFT*: Implica alineamiento a la izquierda.
- *LVCFMT_RIGHT*: Implica alineamiento a la derecha.
- *LVCFMT_CENTER*: Implica alineamiento al centro.

## INSERTANDO NUEVOS ELEMENTOS AL CONTROL

Una vez creado el control y asociadas las listas de imágenes, se pueden insertar los diferentes elementos, y para ello se llama a la función *InsertItem*. En el programa se ha utilizado el siguiente formato de la función:

*int InsertItem( int nItem, LPCTSTR lpszItem, int nImage );*

donde *nItem* es el número de elemento, *lpszItem* es la etiqueta del elemento y *nImage* es el índice de la imagen que se va a asociar con el elemento.

## ESTABLECER LAS VISTAS DEL CONTROL *LIST VIEW*

Algo que parece extraño es que la clase *CListCtrl* no tiene una función para cambiar las vistas que muestran los elementos, y hay que recurrir a las funciones *GetWindowWord* y *SetWindowWord* del API de Windows. Además, al utilizar estas funciones hay que tener cuidado de no eliminar ningún otro atributo anterior, excepto el de la vista que se quiera cambiar. En el siguiente listado se muestra la forma de hacerlo aplicándolo a mostrar la lista de elemento en modo de informe.

```
#define LVS_VIEW (LVS_ICON |
LVS_SMALLICON | LVS_LIST |
LVS_REPORT)
void CListDemoDlg::OnRadioIcon()
{
  DWORD dwStyle;

  dwStyle =
GetWindowLong(m_ListColor.GetSafeH
wnd(), GWL_STYLE);
  dwStyle &= ~LVS_VIEW;

SetWindowLong(m_ListColor.GetSafeH
wnd(),
          GWL_STYLE,(dwStyle |
LVS_REPORT));
  m_ListColor.Arrange(LVA_DEFAULT);
}
```

Como se puede ver, para establecer un nuevo estilo hay que obtener el anterior, quitar todas las vistas, y por último activar el estilo apropiado. La última línea sirve para reorganizar los elementos de la lista.

## EDITANDO LAS ETIQUETAS DE LOS ELEMENTOS

En un control *List View* con el estilo *LVS_EDITLABELS* un usuario puede editar la etiqueta del elemento que está seleccionado. Para ello sólo tiene que hacer click en la etiqueta. Además, desde la aplicación se puede hacer lo mismo llamando a la función miembro *EditLabel*.

Antes de que se comience a editar una etiqueta de un elemento se recibe el mensaje de notificación *LVN_BEGINLABELEDIT*, que permite aceptar o no la edición de las etiquetas de un ele-

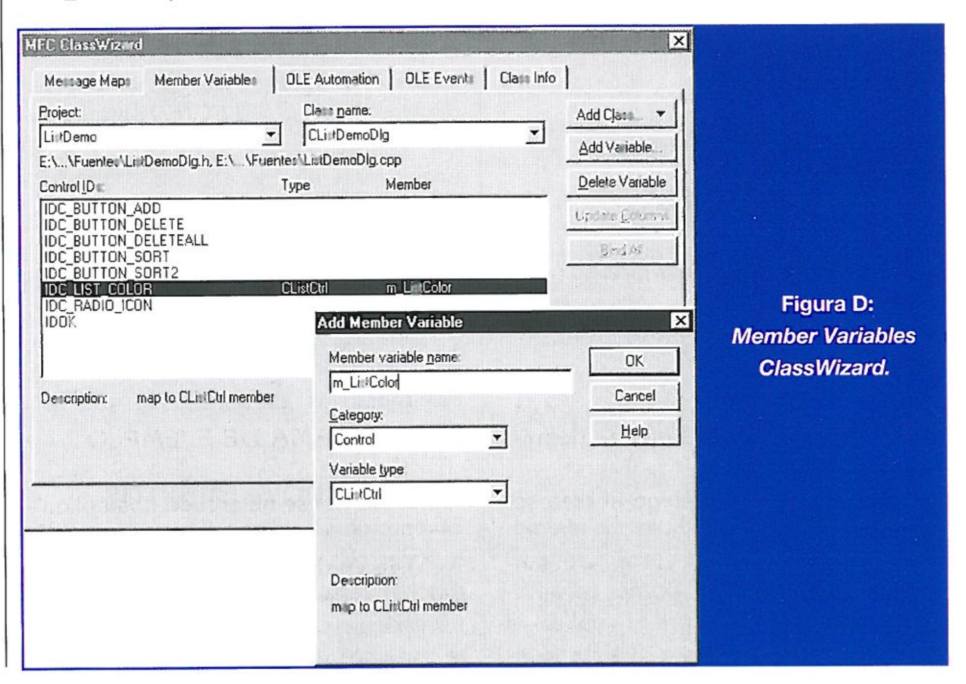



**Figura D: *Member Variables ClassWizard.***

mento. Además, en este momento es cuando se crea el objeto que controla el proceso de edición (la clase *CEdit*), lo que nos permite actuar sobre dicho objeto y por ejemplo limitar la cantidad de caracteres a introducir.

Una vez a finalizada la edición de la etiqueta, se recibe el mensaje de notificación *LVN_ENDLABELEDIT*. Se debe atender a este mensaje si se quiere actualizar la etiqueta del elemento, ya que es responsabilidad del programador validar la cadena introducida, así como de sustituirla en el elemento.

### ELIMINAR ELEMENTOS

Para eliminar un elemento existe la función miembro *DeleteItem*, que recibe el índice del elemento y se encarga de eliminarlo. En el programa de ejemplo se ha combinado esta función con otra que permite conocer si una serie de elementos cumplen una condición, en este caso que estén seleccionados. De esta forma, la función del botón Eliminar permite borrar todos los elementos que en ese momento estén seleccionados. El código que se encarga de esto es el siguiente:

```
void CListDemoDlg::OnButtonDelete()
{
  int Index;
  Index =
m_ListColor.GetNextItem(NO_ITEM,
LVNI_SELECTED);
  while(Index != NO_ITEM)
  {
    m_ListColor.DeleteItem(Index);
    Index =
m_ListColor.GetNextItem(NO_ITEM,
LVNI_SELECTED);
  }
  m_ListColor.Arrange(LVA_DEFAULT);
}
```

La función miembro *GetNextItem* permite obtener los índices de los elementos que cumplen la condición de su segundo parámetro. El primer parámetro indica el elemento por el cual se va a empezar la búsqueda o -1 para localizar el primer elemento que cumpla la condición. Si esta función devuelve -1 quiere decir que no se ha encontrado ningún elemento más que cumpla la condición. Las condiciones son las siguientes:

- *LVNI_ABOVE*: Busca elementos que se encuentren justo encima.
- *LVNI_ALL*: Busca entre todos los elementos.
- *LVNI_BELLOW*: Busca entre todos los elementos que se encuentran justo debajo.
- *LVNI_TOLEFT*: Busca entre todos los elementos justo a la izquierda.
- *LVNI_TORIGHT*: Busca entre todos los elementos justo a la derecha.
- *LVNI_DROPHILITED*: Busca entre todos los elementos que tengan activado el estado *LVIS_DROPHILITED*.
- *LVNI_FOCUSED*: Busca entre todos los elementos que tengan activado el estado *LVIS_FOCUSED*.
- *LVNI_HIDDEN*: Busca entre todos los elementos que tengan activado el estado *LVIS_HIDDEN*.
- *LVNI_MARKED*: Busca entre todos los elementos que tengan activadoel estado *LVIS_MARKED*.
- *LVNI_SELECTED*: Busca entre todos los elementos que tengan activado el estado *LVIS_SELECTED*.

En el programa se ha utilizado el parámetro *LVNI_SELECTED* para localizar los elementos seleccionados. Al encontrarse uno se elimina con la función *DeleteItem*. Esta operación se repite hasta que la función *GetNextItem* devuelve -1, lo que implica que ya no existen más elementos seleccionados.

También existe una función miembro, *DeleteAllItems*, que permite eliminar todos los elementos de una sola vez.

**El control List View tiene dos estilos que permiten ordenar los elementos en función de sus etiquetas**

### ORDENANDO ELEMENTOS

El control *List View* tiene dos estilos que permiten ordenar los elementos en función de sus etiquetas, tanto de forma ascendente como de forma descendente. Además de estas formas de ordenar los elementos se pueden crear nuevos métodos basados en suministrar una función que reciba dos parámetros de dos componentes, los compare e informe de cuál de ellos es el mayor. En el programa se hace la ordenación en función del índice del elemento. Para ello dicho número se guarda en el campo de datos de cada elemento. Se guarda en el campo de datos, ya que este campo se le pasa a la función que se encarga de la ordenación.

Para ordenar los elementos se llama a la función *SortItems*. Esta función tiene como primer parámetro el nombre de la función que debe decidir sobre el orden de dos elementos. El segundo parámetro se le pasa a la función de ordenación. En este caso éste parámetro sirve para indicar si la ordenación es en sentido ascendente o descendente. El formato de la función de comparación es el siguiente:

```
int CALLBACK
CompararElementos(LPARAM lParam1,
LPARAM lParam2, PARAM lParamSoft);
```

En los parámetros *lParam1* y *lParam2* se envían los campos de datos de los elementos a comparar.

### CONCLUSIÓN

Una vez más, se ha pretendido presentar el modo de programación de la clase *CListCtrl* realizando una sencilla aplicación que pretende mostrar algunos de los detalles propios de este control. Por esto, se recomienda al lector que atienda al código fuente del programa, ya que los ejemplos prácticos siempre ilustran mucho mejor la forma de trabajo.

### PRÓXIMO ARTÍCULO

Después de haber mostrado el modo de trabajo con dos de los nuevos controles de Windows 95, se va a dar un giro en el flujo del curso y se entrará en otro de los campos más interesantes de la programación para Windows: la programación del GDI y todo lo que implica, contextos de dispositivos, cursores, iconos, paletas de colores, bitmaps... Esta parte permitirá conocer el modo de introducir gráficos en las aplicaciones.

# EL GESTOR DE VENTANAS X WINDOWS

David Aparicio

Un gestor de ventanas es la parte de un entorno gráfico de usuario (GUI) que dota a éste del aspecto que permite identificarlo y que le da su personalidad. Esto incluye no sólo la forma de las ventanas y posición y estilo de los botones, sino también el comportamiento de los pulsadores del ratón y combinaciones de teclado. En entornos simplificados, así como en las soluciones cerradas, como MS-Windows, la parte del núcleo del entorno y la *shell* del mismo están fuertemente acopladas, y suele no ser posible sustituirlas parcialmente por componentes de terceros. Imaginemos, por ejemplo, un entorno Windows 95 con el núcleo de Windows de 16 bits, o, al contrario, mantener el aspecto visual clásico de Microsoft sustituyendo únicamente el núcleo de 32 bits, simplemente cambiando un par de ficheros. Parece claro que esto no está al alcance del usuario, y que las diferentes capas del sistema van ligadas indisolublemente.

En el entorno X, la parte central está desacoplada de la parte visual. Normalmente, en Linux se proporciona el núcleo de libre distribución del consorcio *Xfree86*, lo que se denomina servidor. La parte gráfica se elige entre varias opciones también gratuitas: *twm*, *olvwm*, ó *fvwm*. Además, existen versiones comerciales de todas estas capas, bien con servidores de casas como MetroLink, así como recubrimientos gráficos compatibles Motif, intercambiables en cualquier combinación. El servidor proporciona la rapidez del entorno, y la parte de *shell* es responsable de su facilidad de uso.

Hoy nos concentraremos en una modificación del *fvwm*, que lo hace semejante en apariencia al tan en boga "Windows 95".

## EL SERVIDOR

En primer lugar, repasaremos el fichero de configuración para el servidor. Esperamos que a estas alturas todos nuestros lectores habituales ya hayan contemplado el entorno gráfico de X en su PC, pero en beneficio a los aficionados más noveles no estará de más repasar este aspecto. Aunque el programa *xf86config* (que se instala con la serie X) permite automatizar la configuración, suele ser útil entender este fichero para modificarlo a mano.

En la tabla 1 se observa un fichero completo, que se localiza en */etc/XF86Config*. Se observa que está estructurado (desde la versión 3.1 del servidor), y se divide en secciones, a saber:

- *Files*: Se especifican las rutas de acceso a ficheros que el servidor usa, actualmente de colores y *fonts*.
- *ServerFlags*: Aquí se indican modificadores globales para el comportamiento del servidor.
- *Keyboard*: Definición de comportamiento de teclas especiales.
- *Pointer*: Definición de comportamiento del puntero, normalmente un ratón.
- *Monitor*: Capacidades físicas y denominación de la pantalla. Puede haber varios.
- *Device*: Descripción del adaptador de vídeo. Puede haber varios.
- *Screen*: Combinación de un monitor, adaptador y resolución junto con un servidor de X.

Todas las secciones siguen una estructura como ésta:

*Section "NombreSección"*
*NombreCampo ValorCampo*
...
*EndSection*

Algunas secciones se pueden repetir a lo largo del fichero de configuración, mientras que otras son únicas. Los




**El entorno X-Windows, debido a la flexibilidad con la que se diseñó, está distribuido en varias capas independientes que pueden sustituirse a voluntad. Hoy se analizará el gestor de ventanas con "*look and feel*" de Windows 95.**

campos suelen ir ligados a secciones concretas. Ahora se echará un pequeño vistazo a cada sección.

## SECCIÓN *FILES*

Tiene dos campos, siendo el primero el de *RgbPath*, que indica el fichero que contiene las etiquetas de los colores que el sistema usa (las combinaciones RGB son de 16 millones, de las que se pueden calificar las más interesantes o representativas). Por ejemplo, a la tupla (0,0,0) se la suele etiquetar como *Black*, para que sea más fácil de recordar. El fichero indicado ántes contiene las etiquetas y los valores RGB que representan. Esto hace posible también variar el valor RGB en este fichero, de modo que se pueda alterar el aspecto de la pantalla aunque los programas sigan utilizando las mismas etiquetas para referenciar los colores que usan.

El segundo campo es *FontPath*, que indica los directorios que contienen familias de fuentes utilizables por el sistema. La familia *misc* es obligatoria, puesto que contiene las fuentes de sistema. Otra familia de uso corriente es la *75dpi*. Como se observa en la figura, este campo se puede repetir tantas veces como directorios de familias indiquemos. La configuración de fuentes en X es un tema interesante, que merecería un artículo aparte.

## SECCIÓN *SERVERFLAGS*

El servidor de X puede usar ciertos parámetros para modificar su comportamiento general, como permitir volcados de depuración (*NoTrapSignals*: recordando que el servidor XFree86 está en constante desarrollo), o habilitar el paso de ciertas combinaciones de teclado a las aplicaciones, o su captura por parte del servidor.

Por ejemplo, la opción *DontZap* hace que la combinación *Ctrl+Alt+BackSpace* finalice la ejecución del servidor. Esto último se usa cuando se están haciendo pruebas de configuración, para minimizar la posibilidad de que se quede pinchada la pantalla. Una vez que la configuración es fiable, deshabilitar esta característica permite que esta combinación de teclas pase a cualquier aplicación que las quiera leer.

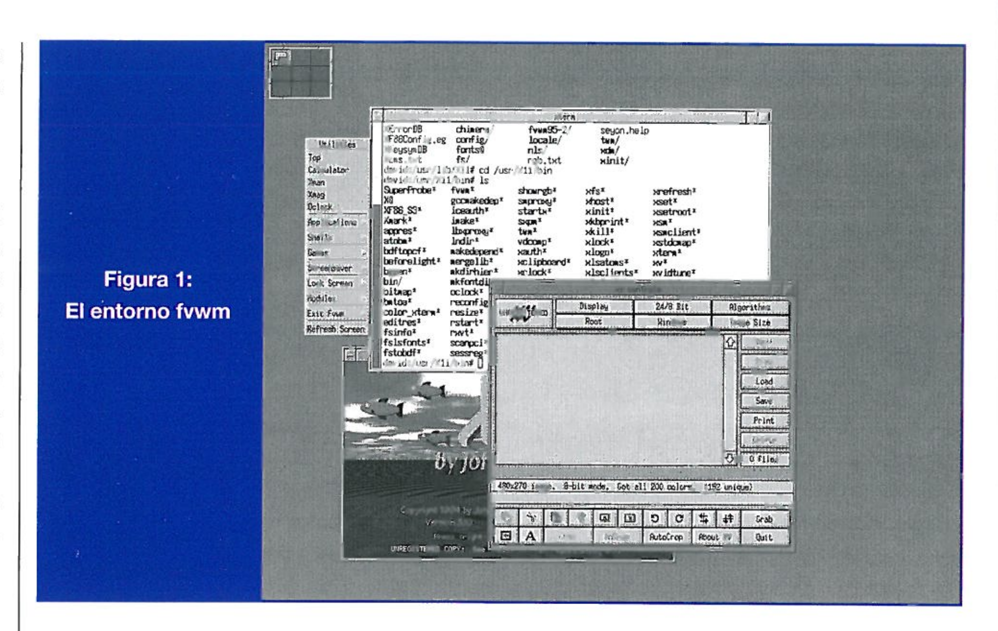


**Figura 1: El entorno fvwm**

## SECCIONES *KEYBOARD* Y *POINTER*

Aquí se definen los protocolos que siguen los dispositivos de entrada, el teclado y el puntero. Se puede controlar si las aplicaciones pueden modificar los leds de teclado (*Xleds*), el mapeo de las teclas físicas (*LeftAlt, RightAlt, RightCtl, ScrollLock...*) con respecto a los modificadores lógicos que el estándar X entiende (*Meta, ModeShift, Compose, ModeLock*). En la configuración del teclado, cada pulsación produce un código numérico que, combinado con los modificadores, resulta en un carácter determinado. Esta matriz se define en otro fichero (*.Xmodmap*, en el subdirectorio */usr/X11/lib/X11/xinit*).

En la configuración de ratón se indica el protocolo (*MouseSystems, Microsoft, PS/2, Logitech...*) y el dispositivo que lo representa (normalmente */dev/mouse*, dependiendo de la configuración).

Además, para ratones de dos botones, es posible emular un tercero pulsando ambos simultáneamente (en X, los ratones de tres botones son más corrientes y útiles, a diferencia de MS-Windows, que imitó el comportamiento de los Macintosh, de un sólo botón, hasta la llegada de Windows 95). La opción que controla esto es *Emulate3Buttons.*

## SECCIÓN *MONITOR*

Aquí llegamos a una sección realmente interesante, y de la que suele depender la buena visualización de las X. Se definen dos campos meramente informativos (*VendorName* y *ModelName*, que representan el nombre del fabricante y del modelo), y otro que permitirá referenciar los datos que se proporcionen aquí, mediante la etiqueta indicada por *Identifier.*

Hay dos campos más que protegen al monitor de posibles desgracias. Los campos *HorizSync* y *VertRefresh* indican frecuencias de barrido que soporta el monitor, y se indican mediante valores discretos, separados por comas, o rangos, separados por guiones (monitores multisíncronos).

Los monitores multisíncronos suelen ser mas flexibles, puesto que se ajustan perfectamente a los parámetros que requiera la tarjeta de vídeo. Poner valores incorrectos en estos dos campos pueden producir roturas irreparables, de modo que se debe acudir al manual del monitor o preguntar a la casa si no se encuentra dicha información.

Por fin, el campo *ModeLine* representa la información más complicada de obtener, la que liga una resolución determinada con unos parámetros de tiempos de barrido y frecuencias a los que la tarjeta debe trabajar. No es tan complicado como pueda parecer, y especialmente para los amantes de "no leer los manuales". Lo único que hay que hacer es copiar la línea correcta de una base de datos para obtener la resolución deseada en base al hardware disponible.

Si el equipo no soporta una entrada determinada, el servidor lo indicará en

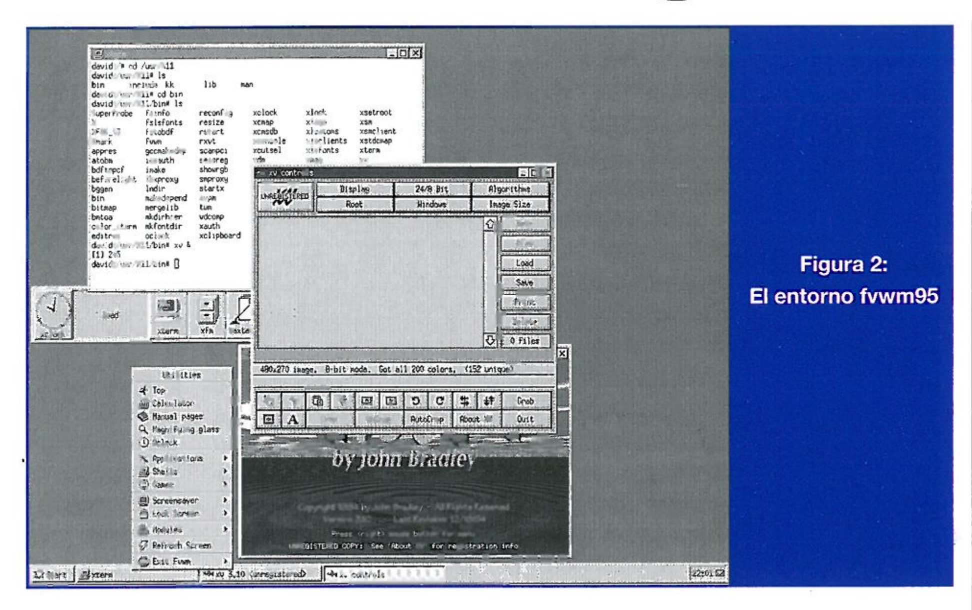


Figura 2:
El entorno fvwm95

el arranque (se recomienda redirigir la salida de errores a un fichero al lanzar las X con *xinit* o *startx*, y examinarlo despues), basándose en los campos de frecuencias soportadas por el monitor. En el ejemplo están habilitadas las líneas estándar para 640x480, 800x600 y 1024x768 para un monitor VESA típico. Las líneas comentadas representan otras combinaciones que pueden ser útiles o no. Recuérdese que sólo puede haber una línea por cada resolución.

## SECCIÓN *DEVICE*

Aquí se configuran características del adaptador gráfico. Al igual que antes, hay dos campos informativos (*VendorName* y *BoardName*) y otro que identifica de forma única a esta combinación (*Identifier*). Tras ello hay otros campos, pero con una entrada como la del ejemplo bastará para que el servidor intente la autodetección del hardware. A diferencia del monitor, cuyas características no pueden ser adivinadas por prueba y error, la placa de vídeo suele ser fácil de testear.

Una recomendación del autor de este artículo es utilizar siempre que se pueda un adaptador de la familia S3, (Trio 64, por ejemplo), que ofrece sencillez de configuración, nitidez, buen rendimiento y bajo coste. Seguro que lo agradecerán a la larga.

## SECCION *SCREEN*

En esta última sección se combina lo anterior, para obtener el resultado final. El campo *Driver* indica el tipo de servidor que está activo (aquel que se haya instalado por defecto de la serie X). A menos que se tengan configuraciones exóticas, lo común es *svga* (para el servidor *XF86_SVGA*) o *accel* (para casi todos los demás). El primero es un servidor común para todas las tarjetas, y el segundo es específico para cada familia, aprovechando las capacidades de aceleración hardware de cada modelo. Para un modelo S3, el segundo es el caso a elegir.

Cuando el servidor arranca, va a buscar en el fichero la sección *Screen* que le corresponde (recordemos que puede haber varias configuraciones), y consulta el resto de los campos, los cuales son *Device* y *Monitor*, para indicar el hardware que se debe emplear para este modo (tarjeta de vídeo y monitor). Excepto para profesionales que disponen de sistemas multimonitor (uno para texto y otro de alta resolución), lo normal es tener un sólo equipo configurado.

Finalmente, se indica una subsección (*Display*) que indica las resoluciones físicas (*Modes*) a emplear, y en qué orden. Estas entradas son etiquetas que referencian líneas *ModeLine* de la sección del monitor.

Por último, mediante las directivas *Depth* y *Virtual* se indica la profundidad de la pantalla (8, 12, 16 o 24 bits) y la resolución lógica a emplear. Si esta última no coincide con la física, la pantalla se desplaza automáticamente cuando se sitúa el puntero en un borde de la misma, consiguiendo un área de visualización mayor de la que permite el monitor, con el único límite de la memoria de vídeo disponible. Por ejemplo, el autor utiliza normalmente una resolución de 800x600 física con 1024x768 lógica en un monitor de 14 pulgadas y tarjeta de 1 Mb de vídeo (con 8 bits de profundidad).

Las profundidades de 16 y 24 bits ofrecen una mayor gama de colores a cambio de una necesidad mayor de velocidad. El usuario notará una pérdida ostensible de rendimiento a menos que su placa de vídeo tenga VRAM instalada.

## EL GESTOR DE VENTANAS

Si se ha instalado la serie X, y tras tener el fichero de configuración correcta, tecleando el comando *startx* se tendrá el clásico sistema de ventanas funcionando (figura 1). Como ya se comentó al principio, es posible sustituir parte de los componentes y alterar el aspecto del sistema. De hecho, el gestor de ventanas (el actual, *fvwm*, así como el que se va a instalar, *fvwm95*) tiene un fichero de configuración aparte e independiente de */etc/XF86Config*. Nos centraremos ahora en este nuevo fichero.

En este artículo se tratará de forma superficial la configuración, y se recomienda utilizar *man fvwm* o *man fvwm95* para consultar todas las opciones disponibles.

Para instalar el gestor con el aspecto Windows 95, bastará con descomprimir el fichero que se proporciona en este número, (el entorno X no debe estar arrancado en ese momento) con los siguientes comandos:

```
cd /
tar xvfz /mi_path/fvwm95.tgz
```

Una vez realizado esto, si se teclea de nuevo *startx*, se comprobarán los cambios realizados, como se observa en la figura 2. Si desea recuperar el gestor antiguo, no se necesita desinstalar nada, y bastará con editar el fichero */usr/X11/lib/X11/xinit/xinitrc*, observando los comentarios del final del mismo.

El fichero de configuración controla aspectos como operaciones de teclado o ratón en las ventanas, la asociación de iconos y cambio de aspecto de las

**CUADRO 1**

```
# Copyright (c) 1994 by The XFree86 Project, Inc.
#
# Refer to the XF86Config(4/5) man page for details

Section "Files"
    RgbPath          "/usr/X11R6/lib/X11/rgb"

    FontPath         "/usr/X11R6/lib/fonts/misc/"
    FontPath
"/usr/X11R6/lib/X11/fonts/75dpi/"
EndSection

Section "ServerFlags"
#   NoTrapSignals
#   DontZap
#   DontZoom
EndSection

Section "Keyboard"
    Protocol         "Standard"
    AutoRepeat       500 5
#   Xleds     1 2 3
#
    LeftAlt     Meta
    RightAlt    ModeShift
#   RightCtl    Compose
#   ScrollLock  ModeLock
EndSection

Section "Pointer"
    Protocol    "MouseSystems"
    Device      "/dev/mouse"
#   Emulate3Buttons
#   Emulate3Timeout    50
EndSection

Section "Monitor"
    Identifier  "CM3209"
    VendorName  "Phillips"
    ModelName   "8CM3209"

    HorizSync   31.5, 48.1, 48.36
    VertRefresh 50-90

# 640x400 @ 70 Hz, 31.5 kHz hsync
Modeline "640x400"    25.175 640  664  760
800  400  409  411  450
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"    25.175 640  664  760
800  480  491  493  525
# 800x600 @ 56 Hz, 35.15 kHz hsync
#ModeLine "800x600"   36     800  824  896
1024  600  601  603  625
# 1024x768 @ 87 Hz interlaced, 35.5 kHz hsync
#Modeline "1024x768"   44.9  1024 1048 1208
1264  768  776  784  817 Interlace

# 640x480 @ 72 Hz, 36.5 kHz hsync
#Modeline "640x480"   31.5   640  680  720
864  480  488  491  521
# 800x600 @ 60 Hz, 37.8 kHz hsync
#Modeline "800x600"   40     800  840  968
1056  600  601  605  628 +hsync +vsync

# 800x600 @ 72 Hz, 48.0 kHz hsync
Modeline "800x600"    50     800  856  976 1040
600  637  643  666 +hsync +vsync
# 1024x768 @ 60 Hz, 48.4 kHz hsync
Modeline "1024x768"   65     1024 1032 1176
1344  768  771  777  806 -hsync -vsync

# 1024x768 @ 70 Hz, 56.5 kHz hsync
#Modeline "1024x768"  75     1024 1048 1184
1328  768  771  777  806 -hsync -vsync
# 1280x1024 @ 87 Hz interlaced, 51 kHz hsync
#Modeline "1280x1024"  80    1280 1296 1512
1568 1024 1025 1037 1165 Interlace

# 1024x768 @ 76 Hz, 62.5 kHz hsync
#Modeline "1024x768"  85     1024 1032 1152
1360  768  784  787  823
# 1280x1024 @ 61 Hz, 64.2 kHz hsync
#Modeline "1280x1024" 110    1280 1328 1512
1712 1024 1025 1028 1054

# 1280x1024 @ 74 Hz, 78.85 kHz hsync
#Modeline "1280x1024" 135    1280 1312 1456
1712 1024 1027 1030 1064

# 1280x1024 @ 76 Hz, 81.13 kHz hsync
#Modeline "1280x1024" 135    1280 1312 1416
1664 1024 1027 1030 1064

EndSection

Section "Device"
    Identifier       "Generic VGA"
    VendorName       "Unknown"
    BoardName        "Unknown"
    Chipset          "generic"
EndSection

Section "Device"
    Identifier  "Ma64"
    VendorName  "ATI"
    BoardName   "Mach64"
#   membase     0xa0000000
#   VideoRam    1024
#   Ramdac      "normal"
#   Clocks  25.18  28.32  31.50  48.36
#   Clocks  12.59  14.16  48.10   0.00
EndSection

Section "Screen"
    Driver      "svga"
    Device      "Ma64"
    Monitor     "CM3209"
    Subsection "Display"
        Depth     8
        #Modes     "1024x768" "800x600"
"640x480"
        ViewPort   0 0
        Virtual    320 200
        #Virtual    1024 768
    EndSubsection
EndSection

# The accelerated servers (S3, Mach32, Mach8,
8514, P9000, AGX, W32, Mach64)

Section "Screen"
    Driver      "accel"
    Device      "Ma64"
    Monitor     "CM3209"
    Subsection "Display"
        Depth     8
        Modes     "800x600" "640x480"
        ViewPort  0 0
        Virtual   1024 768
    EndSubsection
    Subsection "Display"
        Depth     16
        Modes     "640x480" "800x600"
        ViewPort  0 0
        Virtual   800 600
    EndSubsection
    Subsection "Display"
        Depth     32
        Modes     "640x400"
        ViewPort  0 0
        Virtual   640 400
    EndSubsection
EndSection
```

**El fichero de configuración del servidor X**

mismas, color de los componentes y definición de menús y pulsaciones de teclado. Para realizar estas tareas se dispone de:

- funciones internas (*built-in*)
- macros de usuario (*user function*)
- módulos (*plug-ins*)

Las funciones internas se pueden usar en cualquier lugar, y son la base a combinar para realizar cada configuración. Las macros de usuario son agrupaciones de operaciones que se pueden llamar con una etiqueta, de forma transparente. Los módulos son programas externos que sólo se pueden invocar desde el propio gestor, y se comunican mediante tuberías Unix.

Desde el punto de vista de su uso, la referencia a estos tres elementos es indistinguible (a menos que coincidan los nombres, en cuyo caso se debe emplear una sintaxis especial que elimine ambigüedades).

Cuando se inicia el gestor, el fichero de configuración se busca en el directorio base del usuario que lo invoque, con el path *$HOME/.fvwm2rc95*

Si no se encuentra, o contiene errores sintácticos, se intenta leer la configuración global del sistema en la ruta */usr/local/lib/X11/fvwm/.fvwm2rc95*. Si hubiese un nuevo error aquí, el entorno no arrancará correctamente. Con el pulsador izquierdo del ratón se podrá salir del gestor (también se podría abor-

tar con *Ctrl-Alt-Backspace*, si tiene esa opción habilitada en el servidor X). Es práctica habitual dejar un fichero global fijo, por si algo no funciona, y hacer los experimentos de configuración en el fichero del usuario que va a invocar el gestor.

Si se observa la figura 2, se pueden comentar unos cuantos aspectos visuales del entorno. En primer lugar, los botones de la barra superior de cada ventana se distribuyen exactamente de la misma forma que en el nuevo entorno de Microsoft, y sirven (de izquierda a derecha) para menú, minimizar, maximizar y cerrar la ventana. Para mover o cambiar el tamaño de las ventanas se usan las mismas operaciones de ratón, con el pulsador de la izquierda. Todos los iconos (los pequeños que se observan en la esquina superior izquierda de las ventanas, o en los menús, así como los grandes, que se ven al minimizar las ventanas) están en formato XPM, que soporta colores.

Ahora se verán algunos conceptos disponibles en este gestor:

- *SloppyFocus*: Cuando se mueve el ratón por encima de una ventana con campos de entrada, el foco de teclado cambia, sin necesidad de hacer click en dicha ventana. Este foco no se pierde hasta que el ratón pasa por una nueva ventana con capacidad de entrada de teclado.
- *AutoRaise*: Cuando el ratón está durante un tiempo sobre una ventana, ésta pasa a primer plano sin hacer *click* sobre ella.
- *CPP preprocessing*: Es posible utilizar expresiones complicadas en el fichero de configuración (inclusión de ficheros, macros y sustituciones, al igual que las directivas que empiezan por # en un programa C). Revise la página del manual para saber más sobre este aspecto.
- *FvwmPager*: Es posible tener varias pantallas simultáneas (con ventanas asociadas a cada una de ellas) agrupadas en lo que se denomina "entorno de trabajo". Por ejemplo, se pueden tener las aplicaciones asociadas con el desarrollo y depuración en C en un entorno, un juego en otro, una aplicación de música en otro, y así. Es posible cambiar de un entorno a otro con una sola pulsación de ratón, permitiendo así que no se acumulen ventanas en la pantalla.
- *StickyIcons*: Algunas aplicaciones (reloj, calculadora, correo...) pueden necesitarse se esté en el entorno en el que se esté. Para evitar lanzar la misma aplicación una vez por cada entorno, se puede hacer que ésta sea independiente del cambio de entorno, en lo que se llama "pegado al cristal de la pantalla".

## MÓDULOS *PLUG-IN*

En el gestor de ventanas *fvwm* y *fvwm95* existe la capacidad de desarrollar extensiones del gestor sin tener que recompilar el mismo. La comunicación entre estas extensiones y la parte central se realiza mediante tuberías Unix, y para el usuario es como si los módulos formasen parte del gestor, de forma atómica. Esto tiene la ventaja de ahorrar memoria cuando no se están usando dichas funciones, y el inconveniente de un tiempo de respuesta algo inferior, debido a que los componentes se deben arrancar por separado.

Se almacenan juntos en un directorio del sistema, y su ruta se especifica (para que el gestor pueda encontrarlos con sólo referenciar su nombre) con la función interna *ModulePath*. Se incluyen los siguientes por defecto:

- *FvwmAudio*: Asocia sonidos a las operaciones del gestor.
- *FvwmAuto*: Para las funciones de *auto-raising*.
- *FvwmBacker*: Mantiene un fondo de pantalla diferente para cada entorno de trabajo.
- *FvwmBanner*: Muestra un dibujo XPM decorativo. Se suele usar durante el arranque.
- *FvwmButtons*: Muestra una barra de utilidades configurable (como el *LaunchPad* de OS/2).
- *FvwmCpp*: Permite preprocesar el fichero de configuración con cpp.
- *FvwmIdent*: Ofrece información interna sobre una ventana.
- *FvwmM4*: Permite preprocesar el fichero de configuración con m4.
- *FvwmPager*: Este módulo mantiene los diferentes entornos de trabajo.
- *FvwmSave*: Graba la configuración del entorno con formato *.xinitrc*.
- *FvwmSaveDesk*: Graba la configuración con formato *.fvwmrc*.
- *FvwmScroll*: Habilita barras de desplazamiento en cualquier ventana.
- *FvwmTalk*: Para poder introducir comandos de configuración de forma interactiva.
- *FvwmTaskBar*: Una barra de utilidades al estilo de la de Windows 95.
- *FvwmWinList*: Permite obtener una lista de las ventanas activas.

Hay páginas de manual para cada módulo, que se pueden consultar. La configuración se realiza dentro del mismo fichero que el gestor *fvwm95*, con el estilo de los recursos de X-Windows, y se examinará con detalle en un artículo posterior

## FUNCIONES INTERNAS

La mayor parte de la configuración del entorno se basa en estas funciones. No se van a examinar todas, ya que sería algo tedioso y no se utilizarán para una primera configuración básica. Se van a clasificar según su utilidad, considerando:

- Menús y *popups*.
- Definición de funciones.
- Configuración de ventanas.
- Configuración de ratón y teclado.

Algunos aspectos no pertenecen a ninguno de los campos anteriores, y se considerarán de "configuración general". En el cuadro 2 se encuentran estas directivas.

Las líneas *WindowFont*, *IconFont* y *MenuFont* indican el fuente a emplear para el sistema de ventanas, y se puede especificar una definición completa según el estándar X (la primera línea), que es algo engorrosa para uso general. De forma más sencilla, se pueden especificar también las abreviaturas, nombres rápidos para dichas fuentes.

A continuación, se definen los colores de las ventanas inactivas, activas y pegadas. También se pueden usar los nombres descriptivos de los colores, o valores absolutos RGB (recuérdese lo comentado sobre el fichero *rgb.txt*, al hablar de la configuración del servidor).

Cuando se mueve una ventana, *OpaqueMoveSize* indica una ponderación para que el gestor sepa si redibuja toda la ventana durante el desplazamiento, o sólo un área de goma. Con 100 siempre se mueve entera, mientras

que valores menores hacen que las ventanas mas grandes se muevan como áreas de goma. Depende de la potencia de la CPU de que se disponga. *EdgeScroll* y *EdgeResistance* indican el tiempo que debe estar el puntero en un borde de la pantalla para que ésta se desplace (caso de que la resolución virtual sea mayor que la física), y a qué velocidad.

A continuación se define el número de entornos de trabajo disponibles (seis, en el ejemplo con *DeskTopSize*) y luego las rutas para encontrar iconos y módulos. Al especificar estas rutas, en la configuración se pueden omitir los paths absolutos, aumentando la flexibilidad ante posibles cambios.

Finalmente, se definen las propiedades por defecto para todas las ventanas. No se explicarán ahora las opciones que se ven en el ejemplo, puesto que esto pertenece a la configuración de las aplicaciones, que se verán a continuación. Sin embargo, se comentará que el asterisco representa "cualquier ventana", y que esta es la forma de dar opciones por defecto. Si se redefine un recurso para alguna aplicación concreta, se usará la última ocurrencia que haga referencia al mismo. Es decir, que se sigue el orden normal de lectura para saber las propiedades que prevalecen en el fichero de configuración cuando hay ambigüedades.

## DEFINICIÓN DE MENÚS

A continuación se muestra un ejemplo sencillo de menú, mediante el comando *AddToMenu*, que básicamente se compone de título, separadores y opciones. El título se define con la directiva *Title*, mientras que los separadores utilizan la directiva *Nop*. Las opciones se basan en una etiqueta y un comando asociado, normalmente *Exec* para invocar una aplicación, pero también *Popup* para abrir nuevos submenús.

```
AddToMenu ID-Menu
+ "Titulo menu 1" Title
+ "opcion 1" Exec aplic1 parms &
+ "opcion2%icono2.xpm%" Exec aplic2 &
+ ""        Nop
+ "opcion3" Popup        Otro-Menu
```

En este ejemplo se aprecia que se pueden continuar comandos en diferentes líneas, principalmente por claridad, indicando un signo positivo al principio de cada una. Se puede observar en la primera línea que se asocia un identificador al menú que se utilizará para referenciarlo en el resto de la configuración (por ejemplo, en la línea *Popup* se está referenciando a otro menú que debe haber sido definido anteriormente).

La directiva *Exec* es un comando interno que permite lanzar las aplicaciones correspondientes, pasando parámetros de inicialización. Se pone un ampersand al final de la línea para que la nueva aplicación se lance como un proceso independiente.

También aparece en el ejemplo una asociación de un icono (icono2) a una opción de menú (opcion2) , delimitándolo dentro de la etiqueta correspondiente mediante signos de procentaje.

Una vez definidas, estas estructuras se deberán asociar a operaciones de ratón o teclado, en cualquier zona que el usuario desee, como se verá más adelante.

El comando *AddToMenu* no es estructurado, de modo que puede aparecer la definición del menú repartida en varias zonas del fichero de configuración. La primera referencia creará el nuevo menú y el resto son adiciones. Para eliminar el menú se puede usar la función *DestroyMenu*, indicando como parámetro el identificador del que se quiere eliminar.

## FUNCIONES DE USUARIO

En la configuración se pueden definir nuevas directivas que el gestor usará (de forma parecida a macros) transparentemente. La definición es muy sencilla, similar a la que se ha visto en los menús.

```
AddToFunc      ID-Func1
+      "I"     FvwmTaskBar
+      "I"     Exec xsetroot -solid cyan4 &
```

Mediante la función *AddToFunc* se crea o modifica una función de usuario determinada (*ID-Func1*), la cual se puede eliminar en cualquier parte del fichero de configuración con *DestroyFunc*. Para utilizar la nueva función basta indicar su nombre en donde se desee su uso. Si hay ambigüedad con respecto a alguna función interna ya definida, se puede anteponer la palabra clave *Function*. Las siguientes líneas invocan ambas a la función anterior:

*Id-Func1*
*Function Id-Func1*

La estructura de definición de la función es similar a los menús, pero en este caso la etiqueta de la opción se sustituye por una letra que indica la forma en la que se ejecuta cada acción. Se puede expresar:

- I: Acción inmediata. Se realiza en el momento en que se invoca la función.
- M: Arrastre (*Motion*). Se efectúa dicha opción sólo durante una operación de arrastre del ratón.
- C: Pulsación (*Click*). La acción se ejecuta si se ha pulsado y soltado el ratón dentro de un tiempo determinado.
- D: Doble click. La acción se realiza ante una doble pulsación de ratón.

Existen dos funciones especiales, *InitFunction* y *RestartFunction*, que se pueden modificar con la construcción vista anteriormente, y que permiten asociar acciones en el inicio y rearranque del gestor, respectivamente.

## CONFIGURACIÓN DE VENTANAS

Se pueden asociar acciones de la configuración que modifique la apariencia de cada aplicación que se quiera lanzar dentro del entorno. Por ejemplo, se pueden asociar iconos tanto para la esquina superior izquierda de la ventana (*TitleIcon*) como para minimizar la apli-

**El fichero de configuración admite funciones predefinidas, funciones de usuario y módulos externos**

cación (*Icon*). También se pueden decidir cosas como si la ventana tendrá el foco del ratón de forma automática o no (*ClickToFocus*, *MouseFocus*), si está asociada a un entorno de trabajo o no (*Sticky*), y muchas otras opciones. La sintaxis es:

*Style "aplicación" opcion1, opcion2*

Se pueden indicar tantas opciones separadas por comas como se desee, y se pueden indicar varias líneas para una misma aplicación, si la línea de configuración es muy larga. Por ejemplo:

*Style "xterm" Icon xt.xpm,*
*Style "xterm" TitleIcon mxt.xpm*

Adicionalmente, al indicar el nombre de la aplicación se pueden utilizar los comodines "*" y "?" para hacer configuración de grupos de aplicaciones que coincidan en parte del nombre. Como ejemplo, se vio anteriormente que se especifican propiedades por defecto al poner un asterisco como nombre de aplicación, lo que indica "todos los programas".

## EL RATÓN Y EL TECLADO

Esta parte de la configuración es la que va a relacionar todos los menús y funciones que se hayan definido con las operaciones de ratón y teclado que el usuario podrá invocar. La estructura que se sigue es:

*Mouse Dígito Contexto Modifs Acción*
*Key Tecla Contexto Modifs Acción*

El primer caso se usa para configurar el ratón, con un dígito tras la palabra *Mouse* que indica uno de los tres posibles pulsadores (el del centro es el tercero). El segundo sirve para configurar el teclado, indicando *Key* y la tecla a la que se hace referencia, en base a los nombres estándar en una configuración de X-Windows.

El contexto indica la parte de la pantalla en la que aplica cada línea de configuración, y puede ser una combinación de las siguientes letras:

- R: La configuración aplica cuando el puntero está sobre el fondo de la pantalla (background).
- W: El puntero está sobre el marco de una ventana.
- T: El puntero está sobre la barra de título de una ventana.
- S: El puntero está en los extremos de la ventana.
- F: El puntero se encuentra sobre las esquinas de la ventana.
- I: El puntero está sobre el icono de la ventana minimizada.
- 0-9: Se está sobre uno de los botones de la barra de título. Pude haber hasta diez botones, y se numeran de fuera a dentro, con los botones pares a la derecha y los impares a la izquierda (1,3,5,7,9,0,8,6,4,2). En el entorno similar a Windows 95 hay cuatro botones, uno a la izquierda (icono) y tres a la derecha, numerados como 1, 6, 4 y 2.
- A: Cualquier posición excepto los botones de la barra de título.

**CUADRO 2**

```
#Fuentes: para ventanas, iconos y menus
WindowFont        -adobe-helvetica-bold-r-*-*-12-*-*-*-*-*-*-*
IconFont fixed
MenuFont          -adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*

#Colores de las ventanas (color-marco, fondo-marco, color-título, fondo-título)
DefaultColors Black #c0c0c0 #c0c0c0 grey51

#Color de ventana activa y ventanas pegadas (sticky)
HilightColors       White #000080
StickyColors        grey51 #60c080

#Comportamiento al mover las ventanas
OpaqueMoveSize 100
EdgeScroll          100 100
EdgeResistance      10000 0

#Establecer las areas de trabajo
DeskTopSize         3x2

#Rutas para otros componentes del gestor
ModulePath          /usr/lib/X11/fvwm95
PixmapPath          /usr/include/X11/pixmaps:/usr/lib/X11/fvwm95/xpm
IconPath
/usr/include/X11/bitmaps:/usr/lib/X11/fvwm95/xbm

#Propiedades por defecto para todas las ventanas
Style "*" SloppyFocus          # ClickToFocus, MouseFocus o SloppyFocus
Style "*" NoIcon, MWMFuncions, HintOverride, MWMDecor
Style "*" RandomPlacement, DecorateTransient, NoPPosition
Style "*" BorderWidth 5, HandleWidth 5
Style "*" TitleIcon mini-x2.xpm, Icon x2.xpm
```

**Gestor de ventanas: directivas generales**

Los modificadores son las teclas que se pueden pulsar para aumentar las combinaciones posibles de teclado y ratón. Son:

- N: Ningún modificador.
- C: Tecla *Control*
- S: Tecla *Shift*
- M: Tecla *Meta* (normalmente *Alt*)
- A: Cualquier modificador.

Por ejemplo, supongamos que se desea sacar un menú al pulsar el botón derecho del ratón o al pulsar *Alt-F10*, en cualquier parte de la pantalla. Las líneas de configuración serían:

*Mouse 2 A N Popup menu-id*
*Key F10 A M Popup menu-id*

## CONCLUSIONES

Cada vez que los entornos Microsoft y Unix se comparan, inevitablemente aparecen diferencias filosóficas importantes. En el primer caso aparece una simplicidad que, para usuarios expertos es casi desesperante. Dichos entornos están muy integrados, y han sido diseñados para usarlos de una sola forma, con mayor o menor fortuna a la hora de adivinar lo que el usuario quiere.

Por contra, el mundo Unix tiene cara y cruz en la flexibilidad que ofrece. Aunque se puede configurar infinitamente, para usuarios que solo desean "encender y trabajar", el exceso de opciones agobia en las primeras etapas de uso, y hace al sistema poco atractivo.

El entorno *fvwm95* es un ejemplo de cómo se puede mimetizar cualquier aspecto gráfico dentro de X-Windows. Detrás de la apariencia similar se encuentra un fichero de configuración bastante respetable, que obligará a un cierto estudio para empezar a hacer cambios. Sin embargo, la configuración por defecto puede ser conveniente para aquellos que deseen empezar a trabajar sin entrar en complicaciones.

Este artículo pretende ser un primer contacto con las posibilidades de configuración del entorno. Para explotarlo más a fondo, se necesitará examinar las páginas correspondientes del manual. En el futuro, también se explorarán los módulos en un artículo independiente. Hasta otro mes.



# EL SCROLL EN MODO X

*Santiago Romero, Miguel Cubas y Vicente Cubas*

**En el anterior artículo se trató el tema de la memoria en cuanto a su organización. Después de comprender cómo se programan estas dimensiones de pantalla, se entra en el mundo de los scrolls de pantalla, un tema menos teórico a diferencia con los anteriores artículos, y que se podrá poner en práctica para multitud de cosas.**

Con el anterior artículo se terminó de explicar la manera con la que se puede programar la anchura y altura de las pantallas virtuales en Modo X, entendiendo como pantallas virtuales el espacio de memoria direccionable utilizada para guardar los contenidos gráficos. También se trató muy por encima la utilidad que tenia el CRTC en cuanto al scroll y el algoritmo necesario para que este se pudiera realizar. En esta entrega se profundiza en el tema de los scrolles a nivel más profesional.

El registro *Set Start Adress* del CRTC, estudiado en anteriores números, es el encargado de realizar todo aquello que se va a tratar en el presente texto, dedicado integramente a los scrolles. Además se requieren otros registros para adaptar el scroll a lo que el programador desea realizar.

El scroll es un tema que inquieta a la mayoría de los programadores, principalmente a los iniciados en la programación gráfica, debido a su complejidad. Por este motivo, y ya adentrándonos en el Modo X, se va a explicar cómo hacer un scroll en todas direcciones de manera eficaz, sin parpadeos ni diferencias de velocidad y de una forma suave, ideal para realizar demostraciones gráficas o juegos, sistemas que usan muchos juegos de la actualidad. Cuando se vea en el modo 320x200 un scroll tan suave como la velocidad de retrazado, se tratará de un juego realizado en Modo X con estas técnicas.

## EL SCROLL VERTICAL

Este tipo de scroll es el más fácil de realizar y posiblemente el más utilizado por los programadores en sus demostraciones. Un motivo por el que este scroll es más cómodo de utilizar es por la disposición de las pantallas graficas.

Una vez inicializado el modo 13X o X, se presenta automáticamente una organización determinada de las pantallas, que vienen a ser siempre 4, situadas por defecto verticalmente, una encima de otra. Esta es la organización planar que toma por defecto el modo X debido a las rutinas que se utilizan.

Otros motivos por los que el scroll es más facil de realizar así ya se irán conociendo más adelante. Pasemos ahora al scroll vertical en sí.

Una vez volcados los gráficos de la manera adecuada en cada una de las páginas que se tienen en Modo X, es necesario utilizar el registro CRTC y los índices 0Ch y 0Dh para poder desplazar el "marco virtual" de la VGA (lo que se ve) a lo largo de la memoria gráfica (todas las imágenes de que se dispone). Como ya se sabe, estos últimos registros son los encargados de indicar la posición de comienzo a partir de la cual el haz de electrones del monitor tiene que dibujar en la pantalla el mapa de bits (imagen) que se le haya indicado mediante el uso de estos registros(figura 1).

Incrementando continuamente el valor de este registro para avanzar una línea completa se podrá observar un scroll, siempre y cuando antes de llamar a este registro se halla esperado un retrazo vertical, encargado de ralentizar la operación para obtener un scroll tan suave.

Ya que la intención es desplazar la imagen verticalmente, para desplazarla será necesario incrementar el valor 80 unidades cada vez sobre el valor total, que se mandará al registro 0Ch y 0Dh del CRTC. Como ya se sabe, 320 pixels en la pantalla son 80 en Modo X (320

pixels / 4 planos = 80 pixels por plano ), por lo tanto cada vez que se desplacen 80 pixels, la coordenada Y se moverá en la pantalla, es decir se habrá scrolleado el gráfico en el monitor. La manera de utilizar este registro es la siguiente:

```
outportb (CRTC_ADDR , 0x0C);
outportb (CRTC_ADDR+1 , valor >> 8);
/* byte alto */
outportb (CRTC_ADDR , 0x0D);
outportb (CRTC_ADDR+1 , valor &
0xFF);  /* byte bajo */
```

El motivo por el que hay que utilizar dos índices es el siguiente: el valor que se manda al registro de la VGA es un *WORD*, con lo que la única manera de introducirlo es utilizando dos índices que contengan cada uno un byte, ya que los puertos del PC acceden únicamente a bytes, no a *words*.

En el listado 1 se puede ver la manera en la que se puede programar este tipo de scroll. Como ya se ha podido observar es muy sencillo, sólo hay que incrementar la variable en 80 unidades para que se desplace una línea hacia abajo. Esta variable se escribirá en el registro que determina el comienzo de la dirección en memoria cada vez que se incremente, con lo que se obtendrá en pantalla un aparente desplazamiento de la imagen, mientras que lo que se desplaza en realidad es, a modo de ejemplo, el monitor.

**Organización para el scroll vertical**



**FIGURA1.** Pertenece al apartado de "Scroll vertical"

## EL SCROLL HORIZONTAL

Ahora se empieza a complicar el tema con la realización de este tipo de scroll, debido a que existen varias maneras de hacerlo. Antes de empezar con el scroll se tiene que programar la dimensión que va a tener la memoria.

**Los índices 0Ch y 0Dh del CRTC son los encargados de indicar el lugar donde se empezará a volcar la memoria**

Inicialmente las pantallas están organizadas en sentido vertical, y ahora se tienen que poner en horizontal. Como se explicó en los anteriores artículos, el tamaño se programa con el CRTC y el índice 13h (*Offset Register*). Una vez se tengan 4 pantallas seguidas se podrán utilizar los índices 0Ch y 0Dh del CRTC para el scroll. Incrementando en 1 la variable que será escrita en estos últimos registros, se puede observar el desplazamiento horizontal.

Un punto que hay que tener en cuenta siempre que se realice este tipo de scroll es que cuando se incrementa la variable en 1 se desplaza la pantalla en 4 pixels. Esto es debido a los 4 planos del Modo X. El hecho de que se desplace en 4 pixels no indica que el scroll no vaya a ser suave, sino que la velocidad es cuatro veces mayor (y más suave) que si se realizara pixel a pixel. El listado 2 indica la manera de realizar el scroll saltando cada incremento 4 pixels.

Si de lo contrario, se desea realizar el scroll incrementando y desplazando la pantalla en 1 pixel, se tendrá que recurrir al HPP (*Horizontal Pixel Panning Register*), encargado de realizar este trabajo. Este registro se encuentra en el índice 13h del *ATTRIBUTE CONTROLLER* (3C0h), explicado en los artículos

**LISTADO 1**

```
#include <dos.h>
#include "modox.h"

unsigned int x;

void main()
{

Set_320x200X();            /* Programamos Modo
13X */
SelectDimX (40);       /* Organización pantalla 1x4
*/
SetModoActual ( 0 );          /* Modo actual 1x4 */
SetPage(0);            /* Posición página inicial 0 */

LoadImages ();              /* Cargamos imagen */

x = 0;

while (x < 600) {        /* scrolea 3 pantallas */

      VRetrace ();    /* Espera retrazo vertical */

    /* Bloque que escribe en el CRTC para indi-
car */
          /* el Set Start Adress */
asm {
      mov ax , CRTC_ADDR
      mov al , 0Ch
      out dx , al
      mov ax , [x]
      shr ax , 8          /* byte alto del WORD */
      inc dx
      out dx , al
      dec dx
      mov al , 0Dh
      out dx , al
      inc dx
      mov ax , [x]
      and ax , 0FFh        /* byte bajo del WORD */
      out dx , al
  }
      x = x + 80;        /* Incremento 80 que des-
plazará un linea hacia abajo */

   }

asm mov ax , 0003h
asm int 10h

}
```

**Programación de un scroll vertical.**

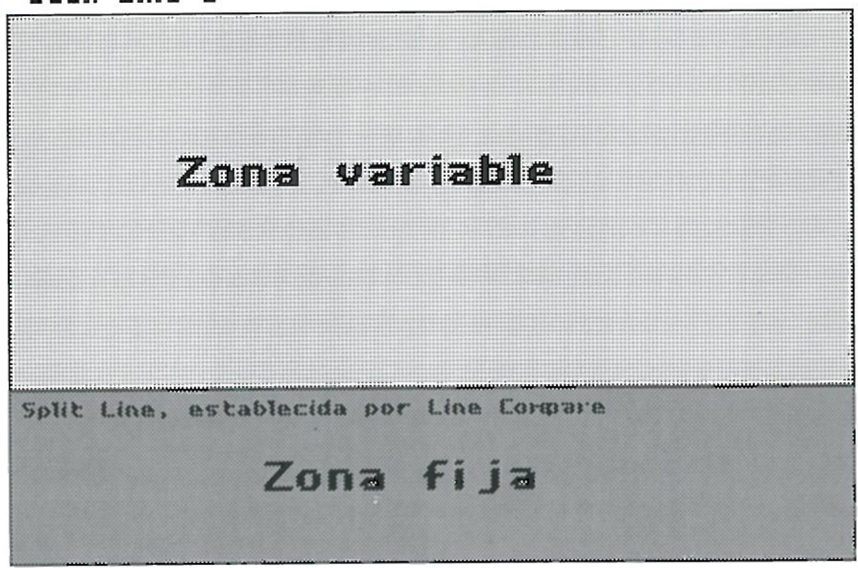



**FIGURA1. Pertenece al apartado de "Scroll vertical"**

que trataban los registros de la VGA, pero este es un tema mucho más complejo de explicar y que requiere mucha experimentación por parte del programador, siendo necesario primero entender todos los sistemas simples de scroll, con los que se pueden hacer efectos como los que hay en los ejemplos del CD.

## EL SCROLL MULTIDIRECCIONAL

Una vez se domine la manera de realizar estos modelos de scroll, se puede efectuar una combinación muy útil para juegos, introducciones o demostraciones gráficas.

Utilizando el índice 0Ch y 0Dh del CRTC, se pueden combinar los scrolls como muestra el listado 3. En los anteriores tipos de scroll se programaba el registro *Set Start Address* una línea más en dirección vertical o bien en dirección horizontal. Para hacer un scroll con trayectoria en diagonal, se debe de calcular el *offset* correspondiente a los valores X e Y (el listado 3 hace esto) y posteriormente mandarlo al puerto e índice correspondiente.

Se pueden crear buenos efectos utilizando el scroll en cuatro direcciones, calculando siempre primero el *offset* donde empezar a dibujar la ventana virtual, y teniendo dibujado dentro de la vídeo memoria algún gráfico contínuo de 640x400 pixels (2x2 pantallas), para realizar un scroll con las cuatro pantallas encadenadas, como en EJEMPLO3.EXE, donde se visualiza un gráfico de este tamaño dentro de la vídeo memoria.

## EL SCROLL CON *SPLIT-SCREEN* "PANTALLA PARTIDA"

Este tipo de scroll, como su nombre indica, determina la pantalla de forma partida, en la cual existe una zona movible y otra estática. La naturaleza de este efecto radica en el desplazamiento de una zona de la pantalla mientras que otra zona se queda fija superpuesta a la zona anterior (siempre aparece por encima). Todo esto por hardware, sin necesidad alguna de rutinas de dibujo. Se tiene así la posibilidad de crear marcadores, ventanas de ayuda e

**El bit 4 del registro 7h es el único que no está protegido por el bit de protección del registro 11h**




**LISTADO 2**

```
Ejemplo 2:

#include <dos.h>
#include "modox.h"

unsigned int x;

void main()
{

Set_320x200X();                /* Programamos Modo
13X */
SelectDimX (160);              /* Organización pantalla
4x1 */
SetModoActual ( 2 );           /* Modo actual 4x1 */
SetPage(0);                    /* Posición página inicial 0 */

LoadImages ();                 /* Cargamos imagen */

x = 0;

while (x < 240) {      /* scrolea 3 pantallas */

      VRetrace ();     /* Espera retrazo vertical */

   /* Bloque que escribe en el CRTC para indi-
car */
          /* el Set Start Adress */
asm {
      mov ax , CRTC_ADDR
      mov al , 0Ch
      out dx , al
      mov ax , [x]
      shr ax , 8            /* byte alto del WORD */
      inc dx
      out dx , al
      dec dx
      mov al , 0Dh
      out dx , al
      inc dx
      mov ax , [x]
      and ax , 0FFh         /* byte bajo del WORD */
      out dx , al
   }
      x++;          /* Incremento en 1*/

   }

asm mov ax , 0003h
asm int 10h

}
```

**Programación de un scroll horizontal.**

información , etc... en medio de la pantalla del programa, juego o demo, como en el caso de *INTROX*, donde se dejan fijas 70 líneas con información en la parte inferior de la pantalla, pudiendo observar en las 130 líneas superiores un desplazamiento de pantalla emulando un paisaje desértico.

El secreto está en el registro *Line Compare o Split-Screen.* Este registro es el encargado de determinar la línea en la cual se divide la pantalla. En la figura 2 se puede observar la estructura de la pantalla en esta modalidad.

El funcionamiento de este registro es muy sencillo: a simple vista se puede ver cómo funciona, viendo que a partir de una determinada línea se para de volcar la memoria, para volver de nuevo al principio de la vídeo memoria (*offset* 0). Pero lo que interesa es cómo funciona el registro internamente, y conociéndolo poder utilizarlo de manera que se puedan crear efectos de calidad profesional que se pueden ver en demos de grupos técnicamente excepcionales.

Internamente la VGA trabaja de forma que, durante la representación de una imagen en pantalla, esta va contando permanentemente la línea actual. En los modos de 200 lineas, como pueden ser el 13h o X, el valor se mueve (a causa de la duplicación de lineas que más adelante se explicará) entre 0 y 400. Este número se encuentra en un registro al cual no se puede acceder a través de puertos, pero mediante otro registro *(Line Compare,* que se encuentra en el CRTC, índice 18h) se puede utilizar para una comparación de línea. La VGA cuenta internamente la línea actual que está dibujando de video memoria a pantalla, mientras este registro compara el valor que contiene con el valor que la VGA cuenta. De este modo cuando el valor es alcanzado, el contador de direcciones se carga con un 0, lo que indica que la representación comienza en esta línea con los datos que se encuentran en el *Offset* 0 de la memoria de vídeo. En pocas palabras, el *Line Compare Register* actúa como un límite en el que la VGA pasa a redibujar a partir del *offset* 0, de manera que se ve un número determinado de líneas de la página actual, y cuando la VGA llega al límite, comienza a redibujar desde el *offset* 0, pudiendo así ver media pantalla de la página 1 y media de la 0 al mismo tiempo que se realiza un scroll.

Dado que el número de línea no se puede guardar en 8 bits (ya que la VGA no reconoce ningún modo con menos de 350 líneas y los modos de 200 líneas son generados por 400 líneas físicas, a esto se debe la duplicación de lineas de la que antes se hablaba), el registro *Line Compare* suele estar dividido en tres registros (que contienen más bits para poder representar valores 0-350),teniendo en las tarjetas Super-VGA hasta cuatro. En el listado 4 se puede observar, para el manejo del *Line*

**LISTADO 3**

```
#include <dos.h>
#include "modox.h"

unsigned int x , y;
char inc_x , inc_y;

void main()
{

Set_320x200X();                /* Programamos Modo
13X */
SelectDimX (160);              /* Organización pantalla
4x1 */
SetModoActual ( 2 );           /* Modo actual 4x1 */
SetPage(0);                    /* Posición página inicial 0 */

LoadImages ();                 /* Cargamos imagen
*/

x = 1;
y = 1;
inc_x = 1;
inc_Y = 1;

while (1 == 1) {       /* condición infinita */

      VRetrace ();     /* Espera retrazo vertical */

   /* Bloque que escribe en el CRTC para indi-
car */
          /* el Set Start Adress */
asm {
      mov ax , CRTC_ADDR
      mov al , 0Ch
      out dx , al
      mov ax , [x]
      shr ax , 8            /* byte alto del WORD */
      inc dx
      out dx , al
      dec dx
      mov al , 0Dh
      out dx , al
      inc dx
      mov ax , [x]
      and ax , 0FFh         /* byte bajo del WORD
*/
      out dx , al
   }
      x = x + inc_x;        /* Incremento en 1 */
      y = y + inc_y;
      if (x <= 0 || x >= 80) inc_x = inc_x * -1;
      if (y <= 0 || y >= 200) inc_y = inc_y * -1;

   asm mov ah , 1    /* comprobación de tecla
*/
   asm int 16h
   jnz Exit

   }

Exit:;
asm mov ax , 0003h
asm int 10h

}
```

**Programación de un scroll multidireccional.**

**LISTADO 4**

```
void SplitScreen ( unsigned char );


void SplitScreen ( unsigned char row )
{
asm {
        mov     bl , row        /* Screen-Splitting en
línea row */
        xor     bh , bh
        shl     bx , 1          /* duplicamos lineas
*/
        mov     cx , bx

        mov     dx , CRTC_ADDR
        mov     al , 07h        /* Registro 7 ( Overflow
Low )   */
        out     dx , al
        inc     dx
        in      al , dx
        and     al , 11101111b  /* Cargar bit 4 con
bit 8 de la línea */
        shr     cx , 4
        and     cl , 16
        or      al , cl
        out     dx , al         /* y fijar              */

        dec     dx
        mov     al , 09h        /* Registro 9 ( Maximum
Row Address ) */
        out     dx , al
        inc     dx
        in      al , dx
        and     al , 10111111b  /* Cargar bit 6 con
bit 9 de la línea */
        shr     bl , 3
        and     bl , 64
        or      al , bl
        out     dx , al         /* y fijar              */

        dec     dx
        mov     al , 18h        /* Registro 18h (Line
Compare / Split Screen) */
        mov     ah , row        /* activar los restan-
tes 8 bits  */
        shl     ah , 1
        out     dx , ax
    }
}
```

**Función que permite programar una Split-Screen.**

**LISTADO 5**

```
Ejemplo 3:


#include <dos.h>
#include "modox.h"
#include "efectx.h"


unsigned int x , y;
char inc_x , inc_y;

void main()
{

Set_320x200X();             /* Programamos Modo
13X */
SelectDimX (160);           /* Organización pantalla
4x1 */
SetModoActual ( 2 );        /* Modo actual 4x1
*/
SetPage(0);                 /* Posición página inicial 0
*/

LoadImages ();              /* Cargamos imagen
*/

x = 1;
y = 1;
inc_x = 1;
inc_Y = 1;

SplitScreen ( 150 );        /* a partir de la línea 150
se encuentra la zona fija */

while (1 == 1) {            /* condición infinita */

      VRetrace ();  /* Espera retrazo vertical */

    /* Bloque que escribe en el CRTC para indi-
car */
            /* el Set Start Adress */
asm {
      mov ax , CRTC_ADDR
      mov al , 0Ch
      out dx , al
      mov ax , [x]
      shr ax , 8            /* byte alto del WORD */
      inc dx
      out dx , al
      dec dx
      mov al , 0Dh
      out dx , al
      inc dx
      mov ax , [x]
      and ax , 0FFh         /* byte bajo del WORD
*/
      out dx , al
   }
      x = x + inc_x;        /* Incremento en 1 */
      y = y + inc_y;
      if (x <= 0 || x >= 80) inc_x = inc_x * -1;
      if (y <= 0 || y >= 200) inc_y = inc_y * -1;

   asm mov ah , 1   /* comprobación de tecla
*/
   asm int 16h
   jnz Exit

   }

Exit:;
asm mov ax , 0003h
asm int 10h

}
```

**Programación de un scroll con Split-Screen.**

*Compare*, la utilización de extensiones de los bits del *Line Compare*, como el *OverFlow Register*.

El listado 4 se encarga de utilizar el *Line Compare* accediendo a dos registros más. Esto se hace de una manera muy sencilla:

El primero de los registros es el *Overflow Register* (CRTC, índice 7). El bit 4 de este mismo registro será el utilizado en este caso, que se encarga concretamente de cargar en su lugar el bit 8 del *Line Compare*. El bit 4 del registro *Overflow* es el único entre los demás que no está protegido por el bit de protección del registro 11h. El bit 9 del registro de comparación de línea se encuentra en el segundo registro que se tiene que utilizar, que es el *Maximum Row Address* (CRTC, registro 9), concretamente en el bit 6. Para programar, por tanto, el *Line Compare* se tendrán que cargar los bits mediante desplazamientos y enmascaramiendos entre estos registros. Este trabajo es realizado por la rutina que se puede observar en el listado 4.

Lo primero que indica el procedimiento de este listado es la duplicación del valor en el modo de 200 líneas. Después se selecciona el *Overflow Register* para cargar en el bit 4 el bit 8 del valor de la línea mediante desplazamiento y enmascaramiento del mismo. Para esto hay que desplazar 4 bits hacia la derecha en el valor de la línea (SHR valor, 4) con tal de dejar el bit 8 en la posición del bit 4.

A continuación se realiza con este último valor obtenido, un AND con 16 para saber si dicho bit(bit 8 del valor, ahora bit 4) está activado o no. Una vez hechos los cálculos se activa o no el bit 4 del *Overflow Register* dependiendo de si lo está o no al hacer el AND con 16.

Estos mismos cálculos se realizan ahora con el bit 9 del la línea y el bit 6 del registro 9h. Finalmente se colocan los restantes 8 bits del número de línea en el registro 18h (*Line Compare* o *Split Screen*).

## PRÓXIMA ENTREGA

En la próxima (y última) entrega de esta serie de artículos se verán rutinas importantes de volcado de *buffers* y *sprites* y visualización de PCXs y rutinas de visualización de GIFs en modo X. Todo un lujo dedicado a las personas que nos han escrito y apoyado con sus comentarios sobre este curso de Modo X. Hasta el mes que viene.

# FUNDAMENTOS DE LA INFORMÁTICA

*José María Peco*

Antiguamente, el hombre trataba la información almacenándola en la memoria humana. Posteriormente, cuando aprendió a leer y escribir, guardó esa información en los libros, pero siguió evolucionando e inventó maquinas para el almacenamiento y posterior tratamiento de esa información. El presente articulo pretende sÓlo ser un pequeño recordatorio de aquellos hitos que han hecho posible el que hoy se pueda hablar de procesos de información mecanizados.

El término más usado para referirse a estos procesos, es el de Informática, siendo ésta una palabra formada por los vocablos Información y Automática, lo cual permite definir Informática como el tratamiento automático de la información. Pero ¿Que es información? Se entiende por Información, en general, el conjunto de datos relativos a hechos, personas y cosas.

Por otra parte, los términos que se emplean para designar las máquinas que procesarán esta información varía en función de la situación geográfica:

- ORDENADOR: de la palabra francesa *Ordenateur* (Europa)
- COMPUTADOR: del ingles *Computer* o contador (Toda América). Lo curioso de estas máquinas es la sencillez conceptual de las operaciones que son capaces de realizar, pues sólo saben realizar las siguientes operaciones básicas:
- Copiar o transcribir la información de un sitio a otro.
- Sumar.
- Restar.
- Elegir entre dos operaciones alternativas.

## ETAPAS DE LA EVOLUCIÓN

En la evolución para el tratamiento de estos datos, se podrían considerar las siguientes etapas:

*Etapa previa.*
Se caracteriza por una evolución lenta de los medios utilizados para la realización de cálculos. Entre estos medios, cabe citar los siguientes:

***Dedos de las manos:*** Basta con fijarse en los niños para entender que el primer medio utilizado para contar fueron los dedos.

***Piedras:*** Este otro medio oermite mantener el resultado del cálculo, liberando a las manos de realizar esta labor. Una reliquia de este medio se puede encontrar en la forma de llevar el tanteo en el juego del MUS.

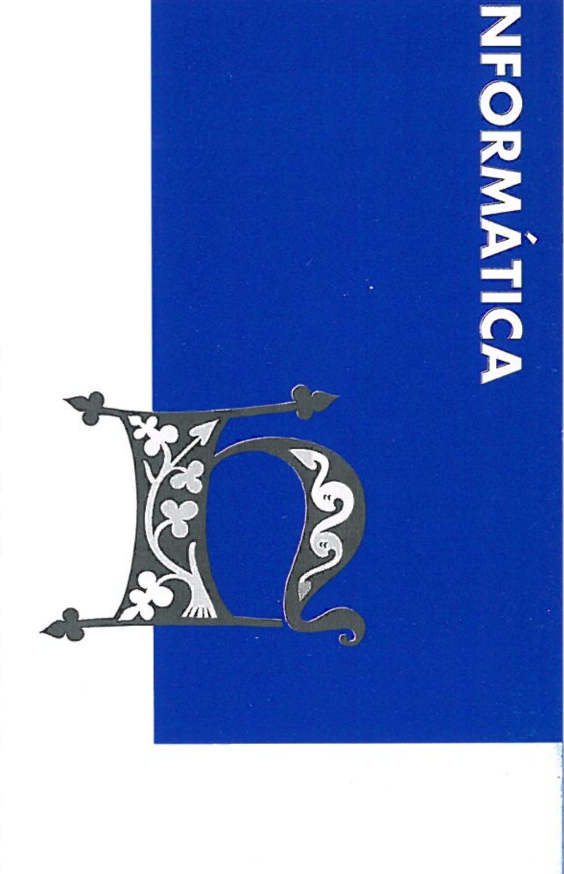
***Ábacos:*** Este medio, de todos conocido, data del siglo IV A. C., y consiste en un conjunto de varillas por las que discurren una serie de bolas o cuentas insertadas en ellas. Los movimientos de estas cuentas irán representando los números, al tiempo que se realizan operaciones aritméticas.

Como dato anecdótico, se puede reseñar que en 1958 se realizó una competición entre dos personas, una de ellas utilizando una calculadora electrónica de la época y la otra utilizando un ábaco. Ganó esta última por 4 a 1.

Un medio similar al ábaco se encuentra en los contadores de bolas chinos, conocidos en el siglo IX A. C.

***Sumadora:*** Es en el siglo XVII cuando aparecen las primeras máquinas de sumar. La primera, construida en 1624 por Wilhem Schikard, se componía de 11 ruedas dentadas completas y de otras 6 ruedas de un único diente, que servían para traspasar el arrastre a la rueda inmediata superior.

La segunda máquina, diseñada por el francés Blas Pascal en 1645, es mucho más complicada y rudimentaria que la de Schikard, por lo que cabe suponer que, aunque de aparición posterior, no llegó a conocerla. Esta



**Durante toda la vida, el hombre ha manejado y procesado datos, pero es en este siglo cuando se ha disparado la generación de datos de una forma exponencial. Por esta razón el hombre, y en especial la industria, ha buscado y potenciado técnicas que ofrezcan seguridad en el proceso de la información.**

máquina consistía en representar cada columna decimal con una rueda dentada con una corona en la que se encontraban inscritos los números del 0 al 9, y otra rueda que permitía traspasar el arrastre a la rueda inmediata superior. La entrada de datos se efectuaba mediante un registrador a estilete (recordar las viejas máquinas registradoras de los comercios), y la lectura de los datos se efectuaba a través de un visor fijo que permitía ver el dígito posicionado tras el mismo.

***Multiplicadora:*** Es también a finales del mismo siglo, en 1694, cuando Libniz construye la primera máquina multiplicadora basada en el principio de adición sucesiva del multiplicando consigo mismo.

Todos estos medios se caracterizan por no ser automáticos, es decir, por necesitar de un operador para su funcionamiento.

***Etapa de desarrollo.***

Esta etapa se puede decir que comienza en 1822, cuando el ingles Charles Babbage, publicó el articulo titulado "*Ejemplos para el cálculo diferencial e integral*", que sirvió de base para la construcción de una máquina que permitía sumar números de seis dígitos, eliminando de esta forma la posibilidad de error humano. Posteriormente comenzó la construcción de otra maquina que fuera capaz de leer datos perforados en un código, pero la muerte le sorprendió sin terminar su investigación.

Esta idea, la lectura de los datos perforados, también fue utilizada por el Francés J.M. Jacquard, el cual, para los telares de su propiedad, diseñó un sistema de fichas perforadas que debían permitir ejecutar secuencias repetitivas de movimientos mecánicos. Pero la inexistencia de una tecnología adecuada impidió que sus ideas pudieran llevarse a la practica.

En 1890 Hermann Hollerith, para poder realizar el 11º censo de la población de los Estados Unidos, creó un equipo de tabulación y estadística basado en fichas perforadas, llegando a construir y utilizar las primeras unidades periféricas electromecánicas. Posteriormente, el doctor Hollerith siguió trabajando en ese tipo de unidades, con lo que perfeccionó el equipo y fundó en 1920 la compañía *TABULATING MACHINE Company* y posteriormente, en 1924, la compañía *INTERNATIONAL BUSINESS MACHINE Company* (IBM).

***Etapa de crecimiento:***

Esta etapa se inicia en 1944, cuando el profesor de Harvard Howard Aiken, en colaboración con IBM, diseña la MARK-I considerada como la primera computadora digital práctica, capaz de ejecutar prolongadas series de problemas aritméticos y lógicos. La información se procesaba a suficiente velocidad y los elementos de entrada y salida eran suficientemente completos, pues los datos se introducían por tarjetas perforadas y los resultados aparecían en formato mecanografiado o perforado. Entre sus características principales se pueden destacar las siguientes:

-Peso: 30 Tm.

-Medidas: 18 m x 2'5 de alto.

-Número de válvulas: 18.000

-Memoria: Sólo podía retener 74 números de 23 cifras.

-Potencia de cálculo: 10 operaciones por segundo.

En 1945, en la Universidad de Pennsylvania, aparece el primer computador electrónico, el ENIAC (*Electronic Numerical Integrator And Calculator*), que aplica las técnicas electrónicas con válvulas de vacío. Entre sus características cuenta con una potencia de calculo 60 veces superior a cualquier otra calculadora conocida.

***Otros hitos de la época son:***

En 1948 se establecen los principios de la Cibernética (piloto automático) como ciencia de las Máquinas de Información.

El siguiente hito se centra en Alemania, con el profesor Von Newman, que establece, junto a otros principios, una filosofía basada en la aritmética binaria.

En 1952, la *Sperry-Rand* diseña la máquina UNIVAC, considerada como el primer paso para el procesamiento de datos automático. Esta calculadora se hizo famosa por pronosticar el triunfo de Eisenhower.

En 1959 comienzan a fabricarse las primeras computadoras transistorizadas, lo cual supone un adelanto importante por la reducción de espacio, aumento de capacidad y reducción de costes.

En 1964 aparecen los circuitos integrados, llamados así por integrar dentro de un mismo *chip* o pastilla varios transistores, capaces de procesar 10^9 instrucciones básicas por segundo

Desde ese momento, la evolución técnica se disparó vertiginosamente, llegando a ser raro el año en el que no se produce una presentación tecnológica que no suponga un avance importante en este campo.

## GENERACIONES DE ORDENADORES

Desde el punto de vista del constructor, se distinguen las siguientes generaciones de ordenadores:

La primera es la generación de los tubos de vacío y relés electromagnéticos, con una densidad de 10 elementos por decímetro cúbico, con la característica consecuente de que al disipar gran cantidad de calor son frecuentes sus averías. Su velocidad de proceso es de 100 operaciones por segundo.

La segunda aparece en 1960 con los transistores y memorias de ferrita, siendo la densidad de elementos de 100 por decímetro cúbico, y su potencia de 100.000 operaciones por segundo. Es en esta época cuando aparecen las cintas magnéticas para el almacenamiento de la información externa.

La tercera generación de ordenadores es la que aparece en 1965, utilizando circuitos integrados con varios miles de elementos por centímetro cúbico.

La cuarta generación se caracteriza por la implantación de circuitos de medio y alto nivel de integración (*MSI* y *LSI*).

## CONSIDERACIONES FINALES

Como se puede apreciar por los hitos reseñados, el avance de las técnicas actuales hace que se haya pasado en pocos años de ser maquinas especializadas en el cálculo a ser máquinas de uso general. Bueno sería que algún estudioso del tema ampliara cada una de las etapas enunciadas y nos deleitara en estas páginas con hechos, anécdotas y curiosidades que, al tiempo que aumentaran nuestra cultura informática, nos hicieran olvidar por unos instantes la parte técnica.

# CORREO DEL LECTOR

**En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación.**

## JERARQUÍA DE CLASES

P *Estoy tratando de hacer un programa gráfico con Borland C++ y su librería BGI. Para intentar practicar y aprender un poco, quería hacerlo con objetos, intentando que fueran lo más general posible. Creé un objeto de tipo general, del que derivé un objeto línea, de este uno rectángulo... Quería que mis rectángulos tuvieran la posibilidad de tener aspecto 3D, para ello intenté utilizar objetos línea, pero no lo conseguí.*

*¿Es posible en un objeto, en mi caso un rectángulo, utilizar otro del que deriva, en mi caso una línea?. Esto es, dentro de un método de rectángulo que generaría el aspecto 3D, definirme un objeto línea que conformaría el deseado aspecto. ¿Cuál es la forma más profesional de hacer esto?*

*José María Zamora Bausá (Madrid)*

R La respuesta es sí. Es posible utilizar dentro de una clase derivada objetos de la clase de la que deriva. Piense en el mecanismo de jerarquía de clases de C++ y en el modo de funcionamiento de la herencia. La creación de una instancia de una clase dentro de una jerarquía implica la creación en memoria de todos los componentes necesarios para cada una de las clases que aparecen en el camino desde la raíz de la jerarquía hasta la clase afectada. No se olvide que una relación de herencia entre clases significa que cada clase derivada incluye una parte de la clase de la que deriva. En su caso, y para explicarlo lo más gráficamente posible (aunque de forma poco rigurosa), si crea cuatro objetos línea dentro de rectángulo, al crear un objeto de tipo rectángulo se crea una estructura de la forma "general + línea + rectángulo". Y este rectángulo contendrá cuatro estructuras de la forma "general + línea".

En su caso, derivar rectángulo de línea no es una buena idea. Piense en la relación de los objetos en la vida real. Un rectángulo contiene líneas, pero no es una línea. Por ejemplo, de una clase base abstracta animal se puede derivar (existe una relación es-un) la clase mamífero, que contendrá los atributos comunes a todos los mamíferos, y de mamífero se puede derivar persona. De este modo, persona tiene todas las características de los mamíferos y de los animales, además de las suyas propias.

Simplificando mucho, la estructura jerárquica que debe utilizar sería la siguiente:

```
class figura {
  private:
    int cx, cy;
    int color;
  public:
    figura ();
    virtual void dibujar () = 0;
};

class linea : public figura {
  private:
    int fin_x, fin_y;
  public:
    linea ();
    void dibujar ();
};

class rectangulo : public figura {
  private:
    linea sup, inf, der, izq;
  public:
    rectangulo ();
    void dibujar ();
};
```

La clase figura es abstracta (no se puede instanciar) porque no tiene sentido crear un objeto figura, pero sí un objeto línea o un objeto rectángulo.

## DUDAS SOBRE DELPHI

P *En primer lugar deseo felicitar a todo el equipo de la revista por el trabajo que realizan a diario con el fin de mostrar todo el material necesario para que este mundo tan apasionante de la programación no sea sólo de los que estudiamos informática, sino también para mucha gente la cual contribuye al diseño de aplicaciones. Mi comentario es si podéis ayudarme a conseguir alguien que haya trabajado o que esté trabajando con Delphi para preguntarle dudas acerca del lenguaje. También listas de debate de Internet que se dediquen a la programación.*

*Humberto Javier Álamo Deniz (Las Palmas de Gran Canaria)*

R Amigo Humberto, si lo que realmente necesitas es resolver dudas sobre Delphi, no tienes más que dirigirte a esta misma sección de la revista. No esperes más y envíanos tus dudas. Por otro lado nos pides listas de debate en Internet. No estamos seguros de si te refieres a foros de debate o grupos de noticias (newsgroups), o a listas de correo, así que te damos información sobre ambas:

En Liszt, un buscador especializado en mailing lists hemos hecho una sencilla búsqueda y nos han aparecido multitud de listas dedicadas al Borland Delphi. El URL de este buscador es:

http://www.liszt.com

En cuanto a grupos de noticias, dispones de los siguientes:

alt.comp.lang.borland-delphi
alt.lang.delphi
alt.online-service.delphi
comp.lang.pascal.delphi.advocacy
comp.lang.pascal.delphi.announce
comp.lang.pascal.delphi.components.misc
comp.lang.pascal.delphi.components.usage
comp.lang.pascal.delphi.components.writing
comp.lang.pascal.delphi.databases
comp.lang.pascal.delphi.misc
es.comp.lenguajes.delphi

Esta última puede ser la que más te interese, pues se habla en castellano.

## LENGUAJE C++ Y POO

**P** ***He comenzado a programar en lenguaje C++ con el compilador de Borland 3.1 y tengo la versión 4.0. Todo mi conocimiento en programación lo obtuve por mis propios medios. Yo les pediría que me informen de todo lo relativo a este lenguaje. Me refiero a bibliografía, cursos, etc. Y a ser posible, de todo lo referido a orientación a objetos.***

*Edgardo Darío Cometto*
*(Provincia de Santa Fe / Argentina)*

**R** Un buen libro para iniciarse en la programación orientada a objetos es "C++. Guía de autoenseñanza" de Herbert Schildt, publicado por McGraw-Hill. Aunque si ya se defiende con el lenguaje, el libro de lectura obligada es "El lenguaje de programación C++" de Bjarne Stroustrup, editado por Addison-Wesley Iberoamericana.

Teniendo en cuenta los compiladores de los que dispone, puede serle útil el libro titulado "Programación orientada a objetos con Borland C++" de Francisco Charte, publicado por Anaya Multimedia. Oros libros interesantes son "Manual de Borland C++ 4.0" de Pappas y Murray, y "The Borland C++ 4.0 Primer" de Weiskamp, del que desconocemos si está disponible alguna edición en castellano.

En cuanto a cursos, Sólo Programadores comenzó un curso de C++ en su primer números, curso que finalizó en el número 14 de la revista. Si lo desea, puede pedir los números atrasados a la propia editorial.

## LÍNEA DE COMANDOS DESDE C

**P** ***Hola, amigos de Sólo Programadores. Soy el típico aficionado a la informática al que le picó el gusanillo de la programación y se decidió a aprender Basic y Pascal. Después de hacer unos cuantos programas me he atrevido a comenzar con C y me he encontrado con un pequeño problema. Quiero hacer un programa al que se le puedan pasar parámetros desde la línea de comandos, pero no encuentro por ningún lado la forma de hacerlo con C. ¿Podrían ayudarme?***

*Oscar Cuadrado Sanz*
*(Burgos)*

**R** El procesamiento de la línea de comandos desde lenguaje C es realmente sencillo. El problema surge en los principiantes por el trato especial de la función main que se hace en C. Si se especifica esta función sin parámetros no hay ningún problema, por lo que se busca alguna librería que contenga funciones especiales para manejar los parámetros pasados en la llamada al programa. Sin embargo, no existe tal librería.

A pesar de las apariencias, la función main admite dos parámetros. El primero es un entero que contiene el número de parámetros pasados en la línea de comandos y el segundo es un vector de cadenas con los parámetros en sí. Siempre existe al menos un parámetro, el número 0, que es el camino completo del nombre del programa. Por esta razón, el entero vale 1 si no hay parámetros o más de 1 si los hay.

A modo de ejemplo, incluimos un pequeño programa que ilustra la utilización de los parámetros de la función main:

```
#include <stdio.h>

int main (int argc, char **argv) {
  int i;
  if (argc == 1)
    printf ("No hay parámetros.\n");
  else {
    for (i=0; i<argc; i++)
         printf ("Parámetro %d: %s\n", i,
argv[i]);
  };
  return 0;
}
```

## SQL Y LENGUAJES EMBEBIDOS

**P** ***Estimados amigos de Sólo Programadores, mis dudas van relacionadas con el mundo de las bases de datos o Sistemas Gestores de Bases de Datos como habitualmente se nombra en vuestra revista a estas aplicaciones. Mi pregunta es la siguiente:¿ qué es un lenguaje embebido? ¿Es útil el SQL? Si lo es, ¿cuándo habrá curso?.***

*Joan Serrano*
*(Reus)*

**R** Empecemos por el final, el mundo de las bases de datos ha sido tratado en diversas ocasiones en nuestra revista (Curso de Clipper, y diversos monográficos sobre paquetes comerciales), pero sí te puedo decir que próximamente se dedicará una serie al lenguaje SQL, el cual es un lenguaje orientado a facilitar la creación, manipulación, protección e integridad de las bases de datos englobadas dentro del modelo relacional, por su sencillez y potencia es hoy en día incluido en todos los paquetes SGBD. Por otra parte hay ocasiones en que es muy útil programar ciertos aspectos de una aplicación en un lenguaje tipo C (aspectos de E/S), y si esa aplicación debe hacer accesos y manipulación de datos, codificar esa parte en SQL, esto es lo que se conoce como un lenguaje embebido en otro.

